ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Мобільний застосунок для автоматизації процесу моделювання під час 3D друку»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Дмитро ЗАДНІЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-42

Дмитро ЗАДНІЧЕНКО

Керівник: Ірина ЩЕРБИНА
канд. тех. наук, доц.

Рецензент: _____

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Задніченку Дмитру Андрійовичу

1. Тема кваліфікаційної роботи: «Мобільний застосунок для автоматизації процесу моделювання під час 3D друку»,
керівник кваліфікаційної роботи канд. тех. наук, доцент Ірина ЩЕРБИНА,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про технології 3D-друку, принципи FDM-друку, формати STL, OBJ, 3MF і G-code, документація з використання Flutter, Dart, Python, FastAPI та SQLite.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі.
2. Проєктування мобільного застосунку.
3. Реалізація програмного продукту.
4. Тестування програмного продукту.

5. Перелік графічного матеріалу: *презентація*

1. Мета, об’єкт та предмет дослідження.
2. Аналіз аналогів.
3. Вимоги до програмного забезпечення.
4. Програмні засоби реалізації.
5. Діаграма варіантів використання.
6. Архітектура програмного продукту.
7. Екранні форми.
8. Тестування програмного продукту.
9. Висновки.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	23.02-28.02.2026	виконано
2	Аналіз та дослідження існуючих аналогів	01.03-07.03.2026	виконано
3	Огляд предметної галузі та технології FDM-друку	08.03-14.03.2026	виконано
4	Проєктування мобільного застосунку	15.03-28.03.2026	виконано
5	Програмна реалізація застосунку	29.03-26.04.2026	виконано
6	Тестування програмного продукту	27.04-04.05.2026	виконано
7	Оформлення роботи: вступ, висновки, реферат	05.05-09.05.2026	виконано
8	Розробка демонстраційних матеріалів	10.05.2026	виконано
9	Попередній захист роботи	11.05-18.05.2026	виконано
10	Подання кваліфікаційної роботи	19.05-30.05.2026	виконано

Здобувач вищої освіти _____ Дмитро ЗАДНІЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЩЕРБИНА
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 92 стор., 14 табл., 29 рис., 21 джерело.

Мета роботи – спрощення процесу створення та підготовки 3D-моделей до FDM-друку за рахунок розробки мобільного застосунку з функціями параметричного моделювання, попереднього перегляду, аналізу друкованості, налаштування параметрів друку, експорту моделей та генерації G-code.

Об'єкт дослідження – процес моделювання та підготовки 3D-моделей до FDM-друку.

Предмет дослідження – програмні засоби автоматизації базового параметричного 3D-моделювання, аналізу друкованості та підготовки моделей до 3D-друку в мобільному застосунку.

Короткий зміст роботи: у роботі проаналізовано процес підготовки 3D-моделей до FDM-друку, особливості пошарового друку, формати STL, OBJ, 3MF та G-code, а також існуючі програмні засоби для моделювання й підготовки моделей до друку. Спроєктовано клієнт-серверну архітектуру мобільного застосунку, структуру бази даних, сценарії використання та інтерфейс користувача. Реалізовано серверну частину на FastAPI, мобільний клієнт на Flutter/Dart, модулі створення параметричних моделей, імпорту STL/OBJ, аналізу друкованості, експорту STL/OBJ/3MF та генерації G-code. Проведено функціональне тестування та перевірку ключових сценаріїв роботи застосунку.

Сферою використання застосунку є навчальні лабораторії, студентські проєкти, початкове прототипування та підготовка простих моделей до FDM-друку.

КЛЮЧОВІ СЛОВА: 3D-ДРУК, FDM, ПАРАМЕТРИЧНЕ МОДЕЛЮВАННЯ, STL, OBJ, 3MF, G-CODE, FASTAPI, FLUTTER, SQLITE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

3D – тривимірний, такий, що має просторове представлення об’єкта;

FDM – Fused Deposition Modeling, технологія пошарового нанесення розплавленого термопластичного матеріалу;

CAD – Computer-Aided Design, система автоматизованого проєктування;

API – Application Programming Interface, програмний інтерфейс взаємодії компонентів;

REST – архітектурний стиль обміну даними між клієнтом і сервером через HTTP;

STL – формат представлення 3D-моделі у вигляді трикутної сітки;

OBJ – формат збереження полігональної 3D-моделі;

3MF – 3D Manufacturing Format, формат збереження даних для 3D-друку;

G-code – набір керуючих команд для 3D-принтера;

SQLite – вбудована реляційна система керування базами даних.

ВСТУП

3D-друк сьогодні використовується не тільки у промисловості, а й у навчальних лабораторіях, гуртках, майстернях, невеликих інженерних проєктах і побутовому прототипуванні. Для студентів та початківців ця технологія є зручною, тому що дає можливість швидко перейти від ідеї до фізичного об'єкта. Водночас сам процес підготовки моделі до друку залишається доволі складним. Користувачу недостатньо просто мати 3D-принтер: потрібно створити або знайти модель, перевірити її розміри, підібрати матеріал, налаштувати параметри друку, оцінити можливі помилки геометрії та сформувати файл, який принтер зможе виконати.

Найпоширенішою для навчального й персонального використання є технологія FDM-друку, що працює за принципом пошарового нанесення розплавленого термопластичного матеріалу. Вона порівняно доступна, зрозуміла для демонстрації та добре підходить для виготовлення простих прототипів, корпусних деталей, навчальних моделей і допоміжних елементів. Проблема полягає в тому, що підготовка навіть нескладної моделі часто потребує використання кількох різних програм. В одному середовищі користувач створює геометрію, в іншому перевіряє або редагує модель, у третьому налаштовує друк і генерує G-code. Для досвідченого інженера такий процес є звичним, але для новачка він може бути незручним і надмірно складним.

Під час роботи з 3D-друком користувач стикається з різними типами програмного забезпечення. CAD-системи дають змогу точно створювати моделі, але потребують розуміння принципів проєктування та часу на навчання. Графічні редактори для 3D-моделювання мають широкі можливості, проте часто орієнтовані на художнє або полігональне моделювання. Slicer-програми, навпаки, добре виконують підготовку до друку, але не призначені для створення повноцінної геометрії з нуля. Через це виникає розрив між створенням моделі та її практичною підготовкою до друку.

Особливо помітною ця проблема є для студентів, які лише починають працювати з FDM-принтерами. На початковому етапі їм не завжди потрібні складні можливості професійних систем на рівні промислового CAD. Частіше необхідно швидко створити просту параметричну модель, змінити її розміри, застосувати базову операцію, перевірити орієнтовні параметри друку та отримати файл для подальшого використання. Саме тому актуальним є створення спрощеного програмного засобу, який об'єднує основні дії в одному зрозумілому сценарії.

Тема бакалаврської роботи пов'язана з розробкою мобільного застосунку для автоматизації процесу моделювання під час 3D-друку. У межах роботи розглядається не промислова CAD-система і не повна заміна професійних slicer-програм, а навчально-прикладний мобільний застосунок, який допомагає виконати базові операції підготовки моделі до FDM-друку. Такий підхід є доцільним для цільової аудиторії, яка потребує простого, доступного та послідовного інструменту для виконання типових дій.

Розроблюваний застосунок орієнтований на роботу з параметричними моделями та базовими геометричними примітивами. У проєкті передбачено створення таких форм, як куб, циліндр, сфера, конус, піраміда та тор, використання готових шаблонів, редагування розмірів, вибір матеріалу й налаштування параметрів друку. Також у системі реалізуються функції аналізу друкованості, попереднього перегляду, імпорту моделей у форматах STL та OBJ, експорту у формати STL, OBJ, 3MF, а також генерації G-code. Саме поєднання цих можливостей в одному мобільному сценарії визначає практичну цінність роботи.

Актуальність теми полягає в необхідності спрощення процесу створення та підготовки 3D-моделей до FDM-друку для користувачів, які не мають достатнього досвіду роботи з професійними інструментами. Наявні програмні рішення здебільшого мають вузьку спеціалізацію або вимагають додаткового навчання. Розробка мобільного застосунку, який поєднує базове параметричне моделювання, аналіз друкованості, розрахунок параметрів друку, попередній перегляд та експорт моделі, дає змогу зробити процес більш послідовним і зрозумілим.

Об'єктом дослідження є процес моделювання та підготовки 3D-моделей до FDM-друку.

Предметом дослідження є програмні засоби автоматизації базового параметричного 3D-моделювання, аналізу друкованості та підготовки моделей до 3D-друку в мобільному застосунку.

Метою бакалаврської роботи є спрощення процесу створення та підготовки 3D-моделей до FDM-друку за рахунок розробки мобільного застосунку з функціями параметричного моделювання, попереднього перегляду, аналізу друкованості, налаштування параметрів друку, експорту моделей та генерації G-code.

Для досягнення поставленої мети необхідно виконати такі задачі:

- проаналізувати предметну галузь 3D-друку та визначити основні етапи підготовки моделей до FDM-друку;
- розглянути існуючі програмні засоби для 3D-моделювання та підготовки моделей до друку, визначити їх переваги й обмеження;
- обґрунтувати актуальність розробки мобільного застосунку для автоматизації базових дій під час моделювання і підготовки до друку;
- сформулювати функціональні та нефункціональні вимоги до програмного продукту;
- спроєктувати архітектуру системи, структуру бази даних, сценарії використання та інтерфейс користувача;
- реалізувати серверну частину застосунку з використанням Python, FastAPI та SQLite;
- реалізувати мобільний клієнт із використанням Flutter та мови програмування Dart;
- реалізувати функції створення параметричних моделей, роботи з шаблонами, виконання операцій над моделлю та аналізу друкованості;
- забезпечити імпорт і експорт моделей у поширених форматах та генерацію G-code для FDM-друку;

- провести тестування розробленого застосунку та перевірити відповідність реалізованих функцій поставленим вимогам.

Для вирішення поставлених задач у роботі використовуються методи аналізу та порівняння, методи системного аналізу, методи моделювання вимог, елементи об'єктно-орієнтованого проєктування, методи проєктування баз даних і методи функціонального тестування. Аналіз і порівняння застосовуються для дослідження існуючих програмних засобів. Системний аналіз використовується для визначення структури мобільного застосунку та взаємодії його основних компонентів. Методи тестування застосовуються для перевірки коректності реалізованих функцій.

Програмний продукт має клієнт-серверну архітектуру. Мобільний клієнт реалізовано за допомогою Flutter та Dart. Серверну частину створено на основі Python і FastAPI. Для збереження даних використано SQLite. Обмін даними між мобільним клієнтом і сервером здійснюється через REST API. У системі передбачено авторизацію користувачів, роботу з 3D-моделями, матеріалами, профілями принтерів, параметрами друку, операціями над моделями, аналізом друкованості та експортом результатів.

Бакалаврська робота складається зі вступу, чотирьох розділів, висновків, переліку посилань і додатка. У першому розділі розглянуто предметну галузь, особливості FDM-друку, формати 3D-моделей і існуючі програмні засоби. У другому розділі наведено проєктування мобільного застосунку, вимоги до системи, архітектуру, структуру бази даних та інтерфейс користувача. У третьому розділі описано реалізацію серверної частини, мобільного клієнта, модулів моделювання, аналізу, імпорту, експорту та генерації G-code. У четвертому розділі наведено тестування розробленого програмного продукту та результати перевірки його функцій.

1 ДОСЛІДЖЕННЯ МЕТОДІВ АВТОМАТИЗАЦІЇ 3D ДРУКУ

1.1. Сутність процесу 3D друку

Адитивне виробництво, до якого належить 3D-друк, є сучасним підходом до виготовлення фізичних об’єктів на основі цифрових моделей шляхом пошарового додавання матеріалу. На відміну від традиційних методів механічної обробки, що ґрунтуються на видаленні надлишкового матеріалу з заготовки, адитивні технології формують виріб безпосередньо з віртуального опису його геометрії. Такий принцип дозволяє значно розширити можливості проєктування складних форм і оптимізувати витрати матеріалів, часу та людських ресурсів [5, 16, 21, 23, 25, 30].

Поява та розвиток адитивного виробництва стали наслідком цифровізації інженерних процесів. Розвиток систем автоматизованого проєктування, комп’ютерного моделювання та числового програмного керування забезпечує перехід від двовимірних креслень до повноцінних тривимірних моделей, які безпосередньо використовуються для виготовлення виробів. У цьому контексті 3D-друк є логічним продовженням цифрового проєктування, де програмна модель виступає основним джерелом даних для фізичного виробництва [8, 12, 21, 23].

Принцип адитивного виробництва полягає у поділі тривимірної моделі на тонкі шари, що послідовно відтворюються у матеріалі. Товщина шару визначає компроміс між швидкістю друку та точністю: менші значення забезпечують кращу деталізацію, але збільшують тривалість процесу. Отже, 3D-друк передбачає оптимальний вибір геометричних і технологічних параметрів [5, 9, 23, 27, 30].

Адитивне виробництво охоплює різні технології, які відрізняються типом матеріалу та способом його затвердіння. Найпоширенішою серед побутових користувачів є технологія пошарового наплавлення термопласту, де матеріал подається через екструдер і формує об’єкт шар за шаром [10, 19]. Також застосовуються фотополімерні методи вибіркового затвердіння рідкого полімеру та порошкові технології спікання або склеювання матеріалу [29].

Незалежно від технології, усі адитивні процеси потребують точного цифрового опису моделі та узгодженої роботи програмного й апаратного забезпечення. Помилки в моделі або алгоритмах її обробки можуть спричинити дефекти виробу, що підкреслює значення програмної складової. Саме на цьому рівні визначаються параметри побудови, траєкторії руху механізмів і режими подачі матеріалу [2, 5. 7, 10].

Важливою перевагою адитивного виробництва є гнучкість: зміна геометрії виробу зводиться до редагування цифрової моделі без переналаштування обладнання. Це створює умови для швидкого прототипування та індивідуалізації продукції, водночас підвищуючи вимоги до якості програмних інструментів [16, 18, 21, 23, 24, 25].

Суттєву роль відіграє параметричний підхід до моделювання, за якого геометрія визначається набором змінних параметрів. Їх зміна автоматично перебудовує модель, що дозволяє створювати сімейства виробів на основі єдиної структури та ефективно інтегрувати 3D-друк із різними програмними платформами, зокрема мобільними застосунками [6, 18, 20, 24].

Загальна структура процесу адитивного виробництва може бути представлена як послідовність взаємопов’язаних етапів, що починаються з формування цифрової моделі та завершуються отриманням готового виробу. На кожному з цих етапів програмне забезпечення відіграє визначальну роль, забезпечуючи перетворення геометричних даних у набір команд, придатних для виконання апаратною частиною 3D-принтера [1, 2, 5, 15, 21].

На рисунку 1.1 подано класифікацію основних процесів адитивного виробництва, у центрі якої відображено загальну категорію «Процеси адитивного виробництва» (Additive Manufacturing Processes) з розгалуженням на ключові технології 3D-друку. Схема охоплює такі методи: струменеве нанесення матеріалу, синтез у шарі порошку, фотополімеризація у ванні, ламінування листових матеріалів, струменеве нанесення сполучного, екструзія матеріалу та пряме підведення енергії та матеріалу. Така схема демонструє роль програмних компонентів у реалізації адитивного підходу [23]

Рис. 1.1 Загальна схема основних процесів адитивного виробництва

З позицій інженерії програмного забезпечення адитивне виробництво слід розглядати як складну інформаційно-технічну систему, у якій програмна логіка безпосередньо визначає поведінку фізичного пристрою. Якість кінцевого виробу залежить не лише від характеристик обладнання чи матеріалу, а й від ефективності алгоритмів нарізання моделі, побудови траєкторій та керування параметрами друку. Тому особливої ваги набувають автоматизація, оптимізація процесів і мінімізація впливу людського фактора [2, 9, 11, 12, 21].

Адитивне виробництво також тісно пов’язане з цифровими екосистемами, оскільки сучасні 3D-принтери інтегруються з мережевими сервісами, хмарними платформами та мобільними пристроями. Це забезпечує можливість віддаленого керування, моніторингу та обміну даними між різними компонентами системи, перетворюючи 3D-друк на складову ширшого програмно-апаратного середовища [4, 6, 26].

На рисунку 1.2 зображено інтерфейс програмного забезпечення для підготовки моделі до друку (слайсер), у якому відображено процес нарізання тривимірної моделі на шари та формування внутрішньої структури заповнення. Ілюстрація демонструє параметри налаштування друку, візуалізацію траєкторій

руху екструдера та попередній перегляд результату, що підкреслює роль програмних модулів як сполучної ланки між цифровою моделлю та її фізичною реалізацією.

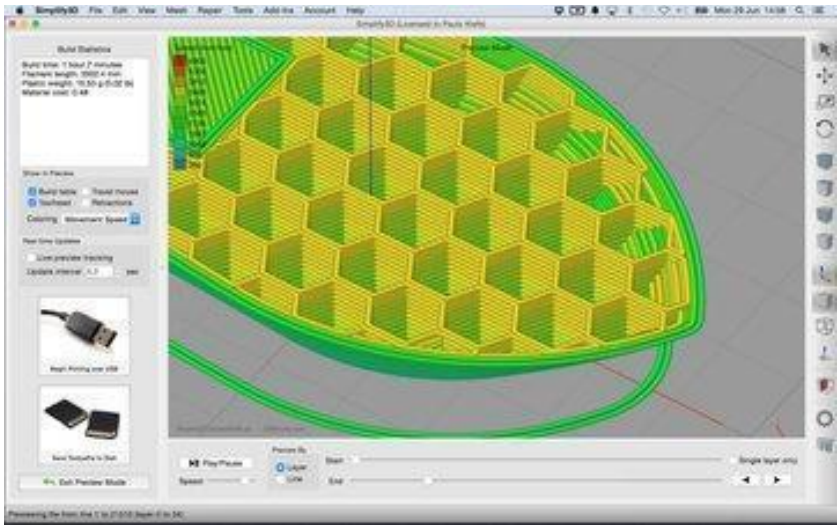


Рис. 1.2 Інтерфейс програмного забезпечення для підготовки моделі до друку (слайсер)

Адитивне виробництво є не лише технологією виготовлення виробів, а й комплексним процесом, у якому програмне забезпечення відіграє ключову роль. Розуміння загальних принципів 3D-друку та адитивного підходу є необхідною основою для подальшого аналізу методів його автоматизації та програмної реалізації, зокрема у вигляді мобільних додатків, орієнтованих на спрощення та оптимізацію процесу моделювання і підготовки до друку [1, 6, 11, 23].

Процес 3D-друку являє собою послідовність взаємопов’язаних етапів, кожен з яких виконує окрему функцію у перетворенні цифрової ідеї на фізичний об’єкт. На відміну від традиційних виробничих процесів, де основні операції зосереджені на механічній обробці матеріалу, у 3D-друці ключову роль відіграє програмна підтримка, яка забезпечує коректність, керованість та відтворюваність усього технологічного циклу [2, 15, 30].

Першим етапом є формування тривимірної цифрової моделі об’єкта. На цьому етапі визначається геометрія майбутнього виробу, його розміри, пропорції та просторові взаємозв’язки між елементами. Модель може створюватися вручну у

системах тривимірного моделювання або генеруватися автоматично на основі параметрів. Незалежно від способу створення, 3D-модель виступає початковим джерелом даних для всіх подальших етапів процесу 3D-друку. Саме тому на цьому етапі важливо забезпечити точність геометричного опису та логічну цілісність структури моделі [8, 18, 20].

Після створення моделі відбувається етап її перевірки та підготовки. Програмне забезпечення аналізує модель на наявність помилок, які можуть унеможливити або ускладнити процес друку. До таких помилок належать незамкнені поверхні, самоперетини, некоректна орієнтація нормалей та надто тонкі елементи, що не відповідають технологічним обмеженням 3D-принтера. Виправлення подібних недоліків вручну вимагає від користувача спеціальних знань і досвіду, тому сучасні програмні системи дедалі частіше використовують автоматизовані алгоритми валідації та корекції геометрії [7, 18, 23].

Наступним важливим етапом є орієнтація моделі у робочому просторі 3D-принтера та налаштування параметрів друку. Вибір орієнтації впливає на якість поверхні, механічну міцність виробу та необхідність використання підтримувальних структур. Програмна підтримка цього етапу полягає у наданні користувачеві інструментів для візуального аналізу моделі, автоматичного визначення оптимального положення та підбору параметрів, що відповідають обраній технології та матеріалу друку [9, 20].

Після завершення підготовчих операцій виконується слайсинг – процес перетворення тривимірної моделі на набір двовимірних шарів. Слайсер аналізує геометрію об’єкта та генерує траєкторії руху робочого інструмента 3D-принтера для кожного шару. У результаті формується керуючий програмний код, який містить інструкції щодо переміщення, швидкості, температури та подачі матеріалу. Якість слайсингу безпосередньо впливає на точність відтворення геометрії та стабільність процесу друку [2, 15, 27].

Програмна підтримка етапу слайсингу є однією з найскладніших складових усього процесу 3D-друку. Вона включає реалізацію алгоритмів розбиття моделі на шари, генерацію внутрішніх заповнень, підтримок та оптимізацію траєкторій руху.

Будь-яка неточність у цих алгоритмах може призвести до дефектів готового виробу або збільшення часу друку. Саме тому сучасні програмні системи активно використовують складні математичні моделі та евристичні методи для досягнення оптимального результату [9, 19].

Після генерації керуючого коду відбувається безпосередній етап фізичного друку. На цьому етапі програмне забезпечення керування 3D-принтером виконує отримані інструкції, координуючи роботу механічних та теплових компонентів пристрою. Система керування відповідає за синхронізацію рухів, підтримання стабільних температурних режимів та контроль стану процесу. У разі виникнення відхилень або помилок програмне забезпечення може призупиняти або коригувати процес друку, що зменшує ймовірність браку [2].

У процесі друку важливу роль відіграє зворотний зв’язок між програмним забезпеченням і апаратною частиною. Дані від сенсорів використовуються для контролю положення, температури та інших параметрів, що дозволяє оперативно реагувати на зміни умов друку. Такий підхід підвищує надійність процесу та забезпечує більш стабільний результат, особливо при тривалих завданнях [23].

Після завершення друку виконується етап після обробки виробу. Він може включати видалення підтримок, очищення поверхні, механічну або термічну обробку. Хоча цей етап не завжди є повністю автоматизованим, програмне забезпечення може надавати рекомендації щодо оптимальних методів після обробки на основі параметрів друку та властивостей використаного матеріалу. Таким чином, програмна підтримка процесу 3D-друку охоплює не лише підготовку та керування, а й інформаційний супровід користувача [22].

Наведений вище ланцюжок етапів дозволяє уявити процес 3D-друку як цілісну систему, у якій кожен етап логічно пов’язаний з попереднім і наступним. Такий підхід є особливо важливим для розуміння місця автоматизації у цьому процесі.

На рисунку 1.3 наведено узагальнену послідовність етапів процесу 3D-друку, починаючи від створення цифрової моделі та завершуючи отриманням готового фізичного виробу. Схема відображає основні стадії програмної підготовки,

генерації керуючого коду та виконання друку, а також демонструє інформаційні зв’язки між програмними і апаратними компонентами системи.

Рис. 1.3 Основні етапи процесу 3D-друку

Розглядаючи процес 3D-друку з позицій інженерії програмного забезпечення, доцільно виділити роль програмних систем як центрального елемента, що координує всі етапи. Саме програмне забезпечення забезпечує узгодженість між геометрією моделі, технологічними обмеженнями принтера та вимогами до якості готового виробу. Це означає, що ефективність усього процесу значною мірою залежить від рівня автоматизації програмних рішень [3].

Особливе значення програмна підтримка має у контексті зменшення впливу людського фактора. Ручне налаштування параметрів, перевірка моделей та підготовка до друку потребують часу і можуть супроводжуватися помилками. Автоматизовані програмні системи дозволяють стандартизувати процес, забезпечити повторюваність результатів та підвищити загальну продуктивність. Це створює передумови для інтеграції 3D-друку у більш широкі програмно-апаратні комплекси, зокрема у мобільні додатки [28].

Етапи процесу 3D-друку утворюють єдиний програмно-керований цикл, у якому автоматизація відіграє ключову роль. Розуміння цього циклу є необхідною основою для подальшого аналізу методів програмної реалізації автоматизованих систем 3D-друку, що розглядаються у наступних підрозділах.

Автоматизація процесу 3D-друку є одним із напрямів розвитку адитивних технологій, що безпосередньо пов’язаний із зменшенням складності підготовки

моделей, підвищенням стабільності результатів та зниженням впливу людського фактора. У традиційному підході до 3D-друку значна частина операцій виконується вручну: користувач самостійно створює або редагує модель, перевіряє її на помилки, налаштовує параметри слайсингу та контролює процес друку. Такий підхід вимагає спеціальних знань, досвіду та часу, що обмежує доступність технології для широкого кола користувачів [12, 18].

З точки зору програмної інженерії автоматизація 3D-друку полягає у перенесенні рутинних і технічно складних операцій з рівня користувача на рівень програмних алгоритмів. Це передбачає створення програмних систем, здатних самостійно аналізувати вхідні дані, приймати рішення щодо оптимальних параметрів друку та генерувати коректний керуючий код без необхідності ручного втручання. У результаті користувач взаємодіє не з низькорівневими технічними налаштуваннями, а з абстрактними параметрами, що описують бажані властивості кінцевого виробу.

Однією з основних складових автоматизації є автоматичне опрацювання геометрії 3D-моделі. Програмні системи виконують аналіз топології об’єкта, виявляють помилки та виконують їх виправлення без участі користувача. Такий підхід дозволяє уникнути ситуацій, коли модель є візуально коректною, але не придатною до друку з технологічної точки зору. Автоматична валідація геометрії забезпечує замкненість поверхонь, коректну орієнтацію нормалей та відповідність мінімальним вимогам до товщини елементів [16].

Важливим моментом автоматизації є параметричне моделювання. Замість роботи зі статичною формою об’єкта користувач оперує набором параметрів, які визначають його геометричні характеристики. Програмна система на основі цих параметрів автоматично формує або змінює тривимірну модель. Такий підхід значно спрощує процес проєктування, оскільки дозволяє швидко адаптувати модель під конкретні вимоги без необхідності повторного ручного моделювання. Параметричне моделювання є особливо ефективним у поєднанні з автоматизованою підготовкою до друку, оскільки забезпечує узгодженість між геометрією об’єкта та технологічними обмеженнями.

Наступним рівнем автоматизації є автоматичний підбір параметрів друку. У традиційних системах користувач змушений самостійно визначати товщину шару, швидкість друку, температуру матеріалу та інші параметри, орієнтуючись на рекомендації або власний досвід. Автоматизовані програмні рішення використовують заздалегідь визначені профілі, алгоритми оптимізації або емпіричні правила для підбору параметрів, що відповідають конкретному типу моделі, матеріалу та принтера. Це дозволяє зменшити кількість помилок і підвищити повторюваність результатів [4, 13].

Суттєву роль у процесі автоматизації відіграє інтеграція етапу слайсингу у загальну програмну архітектуру системи. У сучасних рішеннях слайсинг перестає бути окремим ізольованим інструментом і стає частиною єдиного автоматизованого конвеєра. Програмна система може виконувати слайсинг у фоновому режимі, аналізувати результати та, за необхідності, автоматично коригувати параметри моделі або друку. Такий підхід дозволяє скоротити час підготовки та зменшити навантаження на користувача.

Автоматизація процесу 3D-друку також передбачає використання механізмів контролю та моніторингу. Програмні системи здатні відстежувати стан процесу друку, аналізувати дані від сенсорів та реагувати на відхилення від заданих параметрів. Це створює умови для реалізації адаптивних алгоритмів, які можуть змінювати режими роботи принтера в реальному часі з метою забезпечення стабільності та якості друку. У результаті зменшується кількість зіпсованих виробів і підвищується ефективність використання обладнання [14]
На рисунку 1.4 показано моніторинг виконання завдання. Зображення ілюструє, як програмні модулі ПК та 3D-принтеру передають дані між собою та забезпечують цілісність технологічного процесу.

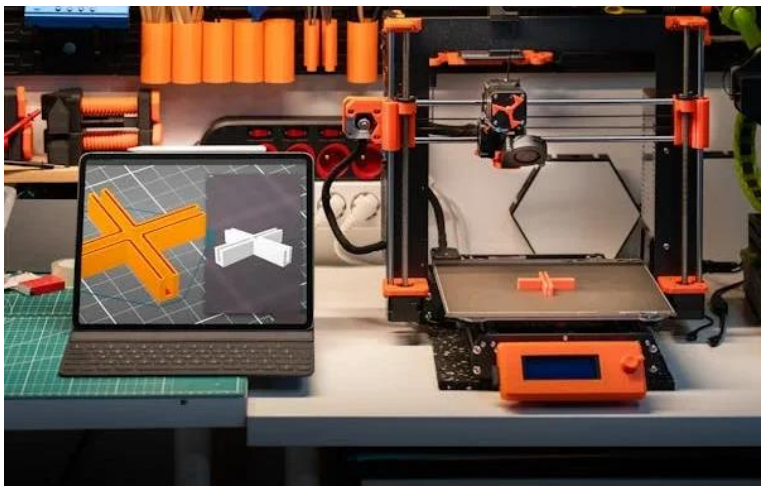


Рис. 1.4 Програмний моніторинг процесу 3D-друку

Особливого значення автоматизація набуває у мобільних програмних систем. Мобільні додатки мають обмежені ресурси у порівнянні з настільними комп’ютерами, що вимагає оптимізації алгоритмів та раціонального розподілу обчислювальних задач. Частина операцій може виконуватися безпосередньо на мобільному пристрої, тоді як більш ресурсоємні обчислення доцільно переносити на серверну або хмарну інфраструктуру. Такий підхід дозволяє поєднати зручність мобільного інтерфейсу з потужністю серверних обчислень.

Автоматизація у мобільних системах також сприяє спрощенню взаємодії користувача з технологією 3D-друку. Інтерфейс мобільного додатку орієнтується на високорівневі сценарії використання, де користувач задає лише основні параметри або вибирає готові шаблони. Усі технічні деталі, пов’язані з підготовкою моделі та генерацією керуючого коду, приховані від користувача і виконуються автоматично. Це робить технологію доступною для користувачів без спеціальної підготовки та знижує поріг входу у сферу адитивного виробництва.

Загалом автоматизацію процесу 3D-друку в програмних системах можна розглядати як перехід від ручного керування окремими етапами до інтегрованого програмного конвеєра, який забезпечує узгодженість, повторюваність та ефективність усього процесу. Такий підхід відповідає сучасним тенденціям розвитку інженерії програмного забезпечення та створює основу для розробки кросплатформенних мобільних рішень.

1.2. Формати програмного коду для 3D друку

Після завершення етапу роботи з геометричними форматами відбувається перехід до формування керуючого програмного коду. Основним форматом такого коду є G-code, який являє собою набір текстових команд, що описують послідовність дій для 3D-принтера. Кожна команда визначає конкретну операцію, зокрема переміщення робочих осей, зміну температури або керування подачею матеріалу. G-code є низькорівневим форматом, тісно пов’язаним з апаратною реалізацією принтера, що забезпечує точне керування процесом друку.

Особливістю G-code є його універсальність та водночас залежність від конкретного обладнання. Хоча загальна структура команд є стандартизованою, окремі виробники 3D-принтерів можуть використовувати власні розширення або інтерпретації команд. Це ускладнює створення універсальних програмних рішень і підвищує значення автоматизованих систем, здатних адаптувати керуючий код під конкретну модель принтера. У цьому контексті важливою є роль слайсерів, які виконують перетворення геометричних даних у G-code з урахуванням апаратних обмежень [2, 26].

Програмні формати керування 3D-друком можуть також включати проміжні представлення, що використовуються для оптимізації або аналізу процесу. Такі формати не призначені для безпосереднього виконання, але слугують для зберігання результатів обчислень, параметрів друку або статистичних даних. Вони є важливими для автоматизованих систем, оскільки дозволяють відстежувати стан процесу та приймати рішення на основі накопиченої інформації.

Важливим моментом використання форматів програмного коду є їх вплив на рівень автоматизації. Формати, що зберігають лише геометричну інформацію, потребують додаткових етапів обробки та ручних налаштувань. Натомість формати, які поєднують геометрію з параметрами друку, створюють умови для реалізації автоматизованих робочих процесів. У мобільних програмних системах

це має особливе значення, оскільки дозволяє зменшити обсяг взаємодії користувача з технічними деталями [10, 20, 21].

З точки зору програмної архітектури формати даних визначають спосіб взаємодії між компонентами системи. Єдиний або узгоджений набір форматів спрощує інтеграцію клієнтських і серверних модулів, забезпечує цілісність даних та зменшує кількість помилок при передачі інформації. Для кросплатформенних мобільних додатків це є критично важливим, оскільки такі системи часто працюють у середовищі з обмеженими ресурсами та нестабільним мережевим з'єднанням.

У таблиці 1.2 наведено порівняльну характеристику основних форматів програмного коду та даних, що використовуються у 3D-друці, з точки зору їх призначення та можливостей.

Таблиця 1.2

Характеристика деяких форматів програмного коду для 3D-друку

Формат	**Призначення**	**Основні можливості**	**Обмеження**
STL	Опис геометрії	Простота, широка підтримка	Відсутність метаданих
AMF	Розширений опис	Матеріали, кольори	Обмежене поширення
3MF	Комплексний контейнер	Геометрія + параметри	Вища складність
G-code	Керування друком	Точне керування	Залежність від принтера

Формати програмного коду для 3D-друку є не просто засобами зберігання інформації, а важливими елементами програмної інфраструктури адитивного виробництва. Вони визначають можливості автоматизації, інтеграції та масштабування програмних систем. Правильний вибір і поєднання форматів дозволяють створювати ефективні програмні рішення, що забезпечують узгодженість між етапами процесу 3D-друку та зменшують кількість ручних операцій.

Вибір оптимального формату файлу для 3D-друку залежить від сукупності технічних, програмних та експлуатаційних чинників. Одним із ключових аспектів є сумісність формату з конкретною моделлю 3D-принтера та програмним забезпеченням, яке використовується на етапах моделювання, слайсингу та керування друком. Навіть за умови формальної підтримки одного й того ж формату різними програмними засобами, особливості його реалізації можуть впливати на коректність обробки геометрії та стабільність процесу друку [2, 21].

Важливу роль відіграє складність тривимірної моделі. Для простих об’єктів із відносно нескладною геометрією доцільно застосовувати формати з мінімальною структурою даних, що забезпечують швидку обробку та високу надійність. У випадку складних моделей із великою кількістю деталей, внутрішніх елементів або параметричних залежностей зростає потреба у форматах, здатних зберігати розширену інформацію без втрати точності та логічної цілісності моделі.

Окремим чинником є необхідність підтримки кольору, текстур і матеріалів. Якщо модель передбачає використання декількох матеріалів або кольорове відтворення, базові формати геометричного опису можуть бути недостатніми. У таких випадках доцільно використовувати формати, які дозволяють зберігати додаткові атрибути та метадані, що істотно розширює можливості автоматизованої підготовки моделей до друку та підвищує якість кінцевого результату.

Не менш важливими є вимоги до точності та рівня деталізації. Формати, що ґрунтуються на полігональному представленні поверхонь, потребують компромісу між геометричною точністю та розміром файлу. Надмірна деталізація призводить до збільшення обсягу даних і часу обробки, тоді як спрощення геометрії може негативно вплинути на якість виробу. Тому вибір формату має враховувати баланс між точністю відтворення моделі та ефективністю її обробки.

Ще одним практичним моментом є розмір файлу та швидкість його обробки. Великі файли вимагають більше обчислювальних ресурсів, часу на передачу та зберігання, що особливо критично для мобільних і хмарних програмних систем. У таких умовах перевагу слід надавати форматам, оптимізованим для швидкої обробки та інтеграції у автоматизовані програмні конвеєри.

Для більшості домашніх користувачів і початківців у сфері 3D-друку формат STL залишається найбільш практичним і універсальним рішенням. Його популярність зумовлена простотою структури, широкою підтримкою у програмному забезпеченні та сумісністю з більшістю побутових 3D-принтерів. Використання STL дозволяє швидко підготувати модель до друку без необхідності врахування додаткових параметрів, що є важливим на початкових етапах роботи з адитивними технологіями [6, 23].

У разі роботи з кольоровими моделями або моделями, що потребують збереження текстур і додаткових атрибутів, доцільно застосовувати формати OBJ або 3MF. Вони забезпечують розширені можливості опису об’єктів і краще підходять для складніших сценаріїв друку та візуалізації. Формат 3MF, зокрема, орієнтований на сучасні автоматизовані робочі процеси та дозволяє зберігати не лише геометрію, а й параметри друку в єдиному контейнері [21].

Для професійного використання та реалізації складних інженерних і промислових проєктів рекомендовано застосовувати формати 3MF, AMF або інженерні формати STEP і IGES. Вибір конкретного формату визначається вимогами до точності, рівня автоматизації, типу обладнання та необхідністю інтеграції з CAD-системами. Такі формати забезпечують збереження геометричної точності, структурованості моделі та додаткової інформації, що є критично важливим для інженерного проєктування.

Таким чином проведений аналіз дозволив сформувати цілісне уявлення про сутність 3D-друку, його ключові етапи та роль програмного забезпечення у забезпеченні ефективності, стабільності й відтворюваності технологічного процесу.

У ході дослідження загальних принципів адитивного виробництва встановлено, що 3D-друк є багаторівневим програмно-керованим процесом, у якому програмне забезпечення виконує центральну роль на всіх стадіях – від створення цифрової моделі до керування апаратною частиною 3D-принтера. Показано, що автоматизація основних етапів підготовки до друку дозволяє

зменшити вплив людського фактора, скоротити час обробки моделей та підвищити повторюваність результатів.

Проаналізовані методи програмування, що застосовуються у системах 3D-друку, засвідчили доцільність використання сучасних підходів інженерії програмного забезпечення, зокрема модульних, кросплатформових та клієнт–серверних архітектур. Встановлено, що такі підходи забезпечують гнучкість програмних систем, спрощують їх масштабування та створюють передумови для реалізації автоматизованих мобільних додатків, орієнтованих на зручність користувача.

Окрему увагу приділено питанням представлення даних і програмного коду, які використовуються у процесі 3D-друку. Показано, що вибір форматів файлів істотно впливає на точність відтворення моделей, ефективність обробки даних та рівень автоматизації підготовки до друку. Узагальнення особливостей основних форматів дозволило обґрунтувати необхідність їх раціонального вибору залежно від складності моделей, вимог до якості та використовуваного програмно-апаратного середовища.

Загалом результати першого розділу підтверджують актуальність і доцільність застосування програмних методів автоматизації для спрощення процесу моделювання та підготовки 3D-друку. Отримані теоретичні положення створюють необхідне підґрунтя для подальшого проєктування й реалізації мобільного додатку автоматизації процесу моделювання 3D-друку, архітектура та функціональні можливості якого будуть розглянуті у наступному розділі роботи.

Формати даних відіграють ключову роль у процесі адаптивного виробництва, оскільки саме вони забезпечується передавання геометричної та технологічної інформації між етапами тривимірного моделювання, програмної підготовки та безпосередньо виготовлення виробу на 3D-принтері. Під час розроблення програмного забезпечення для моделювання та автоматизації процесу 3D-друку враховується, що будь-яка система такого типу функціонує на основі послідовного перетворення даних між різними форматами. Від коректності цього перетворення залежить точність геометрії, відсутність помилок у моделі, а також стабільність

роботи алгоритмів генерації керуючого коду. Таким чином, вибір формату зберігання та обміну тривимірними моделями є важливою складовою архітектури програмної системи. [8].

У загальному вигляді формати, що використовуються у 3D-друці, можна розглядати як проміжні представлення інформації про геометрію об’єкта та параметри його виготовлення. На ранніх етапах процесу домінують формати, орієнтовані на опис геометрії, тоді як на завершальних – формати керуючого програмного коду, призначені для безпосереднього виконання апаратною частиною 3D-принтера. Такий поділ відображає багаторівневу природу адитивного виробництва та підкреслює важливість програмних перетворень.

Найбільш поширеним форматом для представлення тривимірної геометрії є формат STL. У цьому форматі поверхня об’єкта описується у вигляді сукупності трикутних граней, кожна з яких визначається координатами вершин і вектором нормалі. Простота структури STL сприяла його широкому поширенню та підтримці у більшості програмних засобів для 3D-друку. Водночас такий підхід має суттєві обмеження, оскільки формат не містить інформації про одиниці вимірювання, матеріал, колір або ієрархію елементів моделі. У результаті частина важливої інформації втрачається або має відновлюватися на наступних етапах обробки [2].

Обмеження формату STL стали передумовою для розвитку більш сучасних форматів опису 3D-моделей. Формат AMF був розроблений як спроба розширити можливості опису геометрії шляхом додавання підтримки матеріалів, кольорів та складніших структур. Він дозволяє зберігати інформацію про декілька матеріалів у межах однієї моделі та використовує більш гнучкий підхід до опису об’єктів. Проте складніша структура AMF ускладнила його впровадження, що обмежило поширення цього формату у практичних системах.

Подальшим кроком еволюції форматів став формат 3MF, який орієнтований на сучасні вимоги адитивного виробництва. Він дозволяє зберігати не лише геометрію, а й метадані, інформацію про матеріали, параметри друку та взаємозв’язки між елементами моделі. Такий підхід значно полегшує автоматизацію процесу підготовки до друку, оскільки зменшує кількість ручних

налаштувань і забезпечує цілісність даних при передачі між програмними системами. Формат 3MF особливо актуальний для інтегрованих програмних рішень, що орієнтовані на автоматизований робочий процес [10, 21].

Порівняння форматів для 3D друку представлено в таблиці 1.3.

Таблиця 1.3

Порівняння форматів для 3D друку.

Критерії	**STL**	**AMF**	**3MF**	**OBJ**
Тип представлення геометрії	Трикутна сітка	Трикутна сітка	Трикутна сітка	Полігональна сітка
Підтримка матеріалів	Ні	Так	Так	Обмежена(Через MTL)
Підтримка кольору	Ні	Частково	Так	Ні
Метадані та параметри друку	Ні	Так	Так	Ні
Наявність одиниць вимірювання	Ні	Так	Так	Ні
Підтримка декількох об’єктів	Обмежена	Так	Так	Так
Складність структури	Низька	Середня	Висока	Середня
Розмір файли	Середній/великий	Менший (XML-структура)	Оптимізований (архів ZIP)	Середній
Поширеність у FDM-друці	Дуже висока	Низька	Зростаюча	Середня
Простота алгоритмічної обробки	Висока	Середня	Середня	Середня

Як видно з таблиці, формат STL характеризується найпростішою структурою та високою поширеністю у сфері FDM-друку, що забезпечує його сумісність із більшістю CAD-систем і програм для підготовки моделей до друку. Формати AMF і 3MF мають розширені можливості зберігання інформації,

включаючи матеріали, кольори, одиниці вимірювання та параметри друку, проте їхня структура є складнішою, що підвищує вимоги до реалізації програмних модулів обробки. Формат OBJ підтримує полігональну геометрію та кольорові характеристики, однак не містить вбудованих механізмів для збереження параметрів друку і не є спеціалізованим для адитивного виробництва. Також слід враховувати, що більший обсяг метаданих у форматах AMF і 3MF збільшує складність інтеграції та тестування програмних компонентів.

Проаналізувавши наведені характеристики та врахувавши вимоги до розроблюваної системи, мною було обрано формат STL як базовий формат для зберігання та обробки геометрії моделей. Я виходив із того, що основною задачею програми є моделювання та автоматизація процесу 3D-друку з акцентом на надійність, універсальність і простоту реалізації алгоритмів. Використання STL дозволило зосередитися на реалізації механізмів аналізу трикутної сітки, перевірки цілісності моделі та генерації траєкторій без необхідності обробки надлишкових метаданих. Незважаючи на відсутність підтримки матеріалів, кольору та одиниць вимірювання, ці параметри мною були винесені в окремі модулі системи, що забезпечило збереження функціональності без ускладнення структури основного формату. Таким чином, вибір STL є обґрунтованим компромісом між функціональністю, сумісністю та складністю програмної реалізації.

1.3. Вибір методології програмування для автоматизованих систем 3D друку

Процес 3D-друку, є не лише технологічною, але й програмно-орієнтованою системою, у якій ключову роль відіграють методи програмування, що забезпечують автоматизацію, керування та інтеграцію всіх етапів підготовки й виконання друку. На сучасному етапі розвитку адитивних технологій програмне забезпечення виступає центральною ланкою, яка поєднує цифрове моделювання, алгоритмічну обробку геометрії та безпосереднє керування апаратною частиною 3D-принтера [25].

Методи програмування для 3D-друку формувалися поступово, у міру ускладнення самих технологій друку та зростання вимог до точності, надійності й зручності використання. Якщо на початкових етапах розвитку адитивного виробництва програмні рішення обмежувалися генерацією простих керуючих команд, то сучасні системи реалізують багаторівневі архітектури, що включають модулі параметричного моделювання, автоматичної валідації геометрії, оптимізації траєкторій і моніторингу процесу друку в реальному часі.

З точки зору інженерії програмного забезпечення, методи програмування для 3D-друку можна розглядати як сукупність підходів до організації даних, алгоритмів та взаємодії між програмними компонентами. Основною метою цих методів є забезпечення коректного та ефективного перетворення цифрової моделі у фізичний об’єкт з урахуванням технологічних обмежень і характеристик обладнання [19].

Одним із базових підходів є використання процедурного програмування для формування керуючого коду. У цьому випадку програмні алгоритми безпосередньо описують послідовність дій, які має виконати 3D-принтер. Такий підхід історично був першим і досі широко застосовується у вигляді генерації G-code. Програміст або програмна система формує послідовність команд, що керують переміщенням робочих осей, температурними режимами та подачею матеріалу. Перевагою цього підходу є простота реалізації та повний контроль над процесом друку, однак він вимагає глибокого розуміння апаратної частини і не є зручним для автоматизації на високому рівні.

З розвитком систем автоматизованого проєктування поширення набув об’єктноорієнтований підхід до програмування в задачах 3D-друку. У цьому випадку тривимірна модель розглядається як сукупність об’єктів, кожен з яких має власні властивості та методи. Такий підхід дозволяє абстрагуватися від низькорівневих команд і працювати з геометрією на більш високому рівні. Об’єктноорієнтоване програмування спрощує реалізацію параметричних моделей, повторне використання коду та розширення функціональності програмних систем [27].

Важливим напрямом є функціональне програмування, яке використовується для опису геометричних перетворень і побудови складних форм на основі математичних виразів. У функціональному підході модель формується як результат обчислення функцій від заданих параметрів. Це особливо ефективно для параметричного та генеративного моделювання, де зміна одного або кількох параметрів автоматично призводить до перебудови всієї геометрії. Такий метод програмування добре поєднується з автоматизованими системами підготовки до друку та дозволяє реалізувати високий рівень гнучкості [3, 7, 23].

Окрему групу методів становлять скриптові підходи до програмування для 3D-друку. Скриптові мови використовуються для автоматизації типових операцій, керування робочими процесами та інтеграції різних програмних компонентів. Вони дозволяють швидко створювати сценарії обробки моделей, запускати процеси слайсингу та передавати керуючий код на принтер. Скриптові рішення особливо ефективні у середовищах, де необхідно обробляти велику кількість моделей або виконувати повторювані операції.

Зі зростанням складності програмних систем для 3D-друку актуальним став клієнт–серверний підхід. У межах цього підходу частина обчислень виконується на стороні клієнта, а ресурсоємні операції – на сервері. Така архітектура дозволяє використовувати потужні серверні ресурси для складних алгоритмів моделювання та слайсингу, зберігаючи при цьому зручність і мобільність клієнтських додатків. Клієнт–серверна модель є особливо актуальною для мобільних рішень, де обчислювальні ресурси пристрою обмежені [12, 30].

На рисунку 1.5 показано загальну архітектуру програмної системи для 3D-друку, що реалізує клієнт–серверний підхід. Схема ілюструє розподіл функцій між клієнтською частиною, серверними модулями та апаратною частиною 3D-принтера.

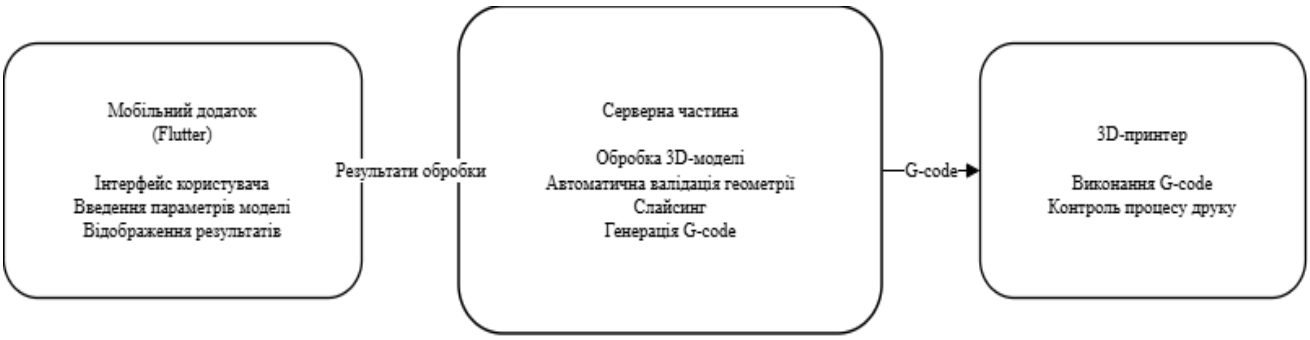


Рис. 1.5 Клієнт–серверна архітектура програмних систем 3D-друку

У межах клієнт–серверної архітектури важливу роль відіграють методи програмування інтерфейсів прикладного програмування. API забезпечують стандартизовану взаємодію між різними компонентами системи, дозволяючи клієнтським додаткам передавати параметри моделей, отримувати результати обробки та керувати процесом друку. Використання API сприяє модульності програмного забезпечення та полегшує його масштабування.

Особливу увагу слід приділити методам програмування, орієнтованим на автоматизацію. Автоматизовані алгоритми приймають рішення на основі аналізу вхідних даних і заданих правил. У задачах 3D-друку це може включати автоматичний вибір параметрів друку, генерацію підтримувальних структур та оптимізацію траєкторій руху. Реалізація таких алгоритмів потребує поєднання геометричних обчислень, евристичних методів та емпіричних правил [11].

У сучасних програмних системах для 3D-друку дедалі частіше застосовуються модульні підходи до програмування. Функціональність системи поділяється на окремі модулі, кожен з яких відповідає за конкретний етап процесу. Така організація коду спрощує тестування, супровід і розвиток програмного забезпечення. Модульний підхід є особливо важливим для складних систем, що включають як клієнтські, так і серверні компоненти [26].

У таблиці 1.3 наведено порівняльну характеристику основних методів програмування, що застосовуються у системах 3D-друку, з точки зору їх переваг та обмежень.

Таблиця 1.3

Порівняння методів програмування для 3D-друку

Метод програмування	**Характеристика**	**Переваги**	**Обмеження**
Процедурний	Генерація послідовних команд	Простота, контроль	Низький рівень абстракції
Об’єктноорієнтований	Робота з геометричними об’єктами	Гнучкість, розширюваність	Вища складність
Функціональний	Опис геометрії через функції	Параметричність	Складність реалізації
Скриптовий	Автоматизація процесів	Швидкість розробки	Обмежена масштабованість

Значну роль у сучасних системах 3D-друку відіграють кросплатформові методи програмування. Вони дозволяють створювати програмні продукти, що працюють на різних операційних системах без необхідності розробки окремих версій. Для мобільних додатків це є особливо важливим, оскільки забезпечує єдину кодову базу та спрощує супровід програмного забезпечення. Кросплатформовий підхід сприяє швидкому впровадженню нових функцій та зменшенню витрат на розробку.

Методи програмування для 3D-друку також тісно пов’язані з питаннями безпеки та надійності. Програмні системи повинні забезпечувати захист даних користувача, коректну передачу керуючого коду та стабільність роботи під час тривалих процесів друку. Це вимагає використання перевірених алгоритмів, механізмів обробки помилок та тестування програмного забезпечення на різних рівнях.

Можна зазначити, що методи програмування для 3D-друку охоплюють широкий спектр підходів – від низькорівневих процедурних рішень до складних клієнт–серверних і кросплатформенних систем. Вибір конкретних методів визначається вимогами до автоматизації, масштабованості та зручності використання програмного забезпечення. Саме ці аспекти є визначальними при розробці мобільних додатків для автоматизації процесу моделювання 3D-друку.

1.4 Висновки за проведеним дослідженням

У результаті проведеного аналізу встановлено, що адитивне виробництво є складною програмно-керованою системою, у якій ефективність процесу 3D-друку визначається якістю програмного забезпечення та узгодженістю всіх етапів — від створення цифрової моделі до фізичного виготовлення виробу. Автоматизація підвищує ефективність, зменшуючи вплив людського фактора, скорочуючи час підготовки та забезпечуючи стабільність і повторюваність результатів.

У ході дослідження форматів даних і програмного коду визначено, що вони є ключовою складовою програмної інфраструктури 3D-друку. Вони перетворюють геометричну інформацію у керуючі команди та впливають на точність, швидкість і автоматизацію; обраний формат STL забезпечує простоту, універсальність і широку підтримку.

Аналіз методів програмування показав, що сучасні системи 3D-друку потребують поєднання кількох підходів. Обрано модульний і об’єктноорієнтований підходи як основу через їх гнучкість і зручність супроводу, доповнені клієнт–серверною архітектурою для ефективного розподілу ресурсів і підтримки мобільних застосунків.

Таким чином, результати розділу підтверджують доцільність використання STL як основного формату даних та модульної кросплатформової архітектури з елементами клієнт–серверного підходу як бази для розробки програмної системи автоматизації 3D-друку. Це створює теоретичну основу для подальшої розробки мобільного додатку, спрямованого на спрощення моделювання та підготовки до друку.

2 ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

2.1 Загальна характеристика програмного продукту

Розроблюваний програмний продукт є мобільним застосунком, призначеним для автоматизації базових етапів моделювання та підготовки 3D-моделей до FDM-друку. Його основна ідея полягає в тому, щоб об’єднати в одному середовищі ті дії, які користувач зазвичай виконує у кількох різних програмах: створення простої моделі, вибір параметрів друку, попередній аналіз, перегляд результату, експорт моделі та формування G-code.

Застосунок розробляється не як промислова CAD-система і не як повноцінна заміна професійних slicer-програм. Його доцільно розглядати як навчально-прикладний інструмент для студентів, початківців у сфері 3D-друку та користувачів навчальних FDM-принтерів. Такий підхід дозволяє зосередитися не на складному інженерному проєктуванні, а на зрозумілому проходженні базового сценарію: створити модель, налаштувати друк, оцінити результат і підготувати файл для подальшого використання.

Цільова аудиторія застосунку включає користувачів, які вже мають базовий інтерес до 3D-друку, але ще не володіють професійними CAD-системами або не хочуть використовувати кілька різних програм для виконання простої задачі. Для таких користувачів важливими є простий інтерфейс, зрозуміла послідовність дій, наявність готових шаблонів, можливість швидко змінити параметри моделі та отримати попередню оцінку друку. Саме тому функціональність програмного продукту орієнтована на практичний і навчальний сценарій, а не на надмірно деталізоване професійне моделювання.

Основний робочий процес у застосунку починається з авторизації користувача. Після входу в систему користувач отримує доступ до списку моделей, dashboard-модуля, шаблонів, створення нової моделі, імпорту файлів та перегляду деталей уже збережених моделей. Така структура дозволяє організувати роботу з моделями не як випадковий набір функцій, а як послідовний процес, у якому кожен екран відповідає певному етапу підготовки до друку.

Однією з ключових функцій застосунку є створення параметричних моделей. Користувач може обрати тип базового геометричного об’єкта та задати його параметри. У межах програмного продукту передбачена робота з такими примітивами, як куб, циліндр, сфера, конус, піраміда та тор. Такий набір є достатнім для навчальних і базових прикладних задач, оскільки дозволяє швидко продемонструвати принцип створення тривимірної форми, вплив розмірів на об’єм і подальший зв’язок моделі з параметрами друку.

Крім створення моделі з нуля, застосунок підтримує роботу з шаблонами. Шаблони потрібні для того, щоб користувач міг швидко створити типову модель без повторного ручного введення всіх параметрів. Такий підхід є зручним для навчального процесу, коли потрібно часто працювати з однаковими або схожими моделями, наприклад калібрувальним кубом, технічним корпусом, простою деталлю або демонстраційною формою. Використання шаблонів скорочує кількість дій користувача та робить роботу із застосунком більш практичною.

Важливою функцією є імпорт готових моделей у форматах STL та OBJ. Це розширює сценарії використання застосунку, оскільки користувач може працювати не тільки з моделями, створеними безпосередньо в системі, а й із файлами, отриманими з інших програм або джерел. Імпортована модель може бути використана для перегляду, аналізу та подальшої підготовки. Таким чином, застосунок не обмежує користувача лише вбудованими параметричними формами.

Для підготовки до друку у програмному продукті передбачено роботу з матеріалами та профілями принтерів. Користувач може обрати матеріал, зокрема PLA, ABS або PETG, а також налаштувати параметри друку: висоту шару, відсоток заповнення, швидкість, температуру сопла, температуру друкованого столу, діаметр сопла та потребу в підтримках. Ці параметри мають практичне значення, оскільки впливають на якість друку, міцність виробу, час виготовлення, витрату філаменту та орієнтовну вартість.

Окремим блоком є аналіз друкованості моделі. Його призначення полягає в тому, щоб надати користувачу попередню інформацію до запуску друку. У межах аналізу система розраховує орієнтовний об’єм, площу поверхні, масу матеріалу,

довжину філаменту, час друку та приблизну вартість виготовлення. Для навчально-прикладного застосунку такі показники є важливими, оскільки вони допомагають користувачу зрозуміти зв’язок між геометрією моделі, вибраним матеріалом і параметрами друку.

Застосунок також передбачає попередній перегляд моделі. Візуальна перевірка дозволяє користувачу оцінити форму об’єкта, його пропорції та загальну відповідність очікуваному результату. Для цього використовується спрощений 2D або ізометричний перегляд, а також 3D-представлення моделі. Наявність попереднього перегляду є важливою, оскільки користувач не повинен працювати лише з числовими параметрами без візуального контролю результату.

Після створення, налаштування та аналізу модель може бути експортована. У програмному продукті передбачено підтримку форматів STL, OBJ, 3MF та G-code. Формати STL, OBJ і 3MF використовуються для збереження або передавання геометрії моделі, а G-code є набором керуючих інструкцій для FDM-принтера. Підтримка цих форматів дозволяє застосунку охопити не тільки етап створення моделі, а й частину процесу її підготовки до фізичного друку.

З технічної точки зору програмний продукт побудований за клієнт-серверною архітектурою. Мобільний клієнт реалізується з використанням Flutter та мови Dart. Він відповідає за інтерфейс користувача, навігацію між екранами, введення даних, відображення моделей і результатів аналізу. Серверна частина реалізується з використанням Python та FastAPI. Вона відповідає за обробку запитів, роботу з моделями, виконання розрахунків, експорт файлів, генерацію G-code та взаємодію з базою даних SQLite. Такий поділ дозволяє відокремити інтерфейсну частину від логіки обробки даних.

Основні характеристики програмного продукту наведено в таблиці 2.1.

Таблиця 2.1

Загальна характеристика розробленого мобільного застосунку

Характеристика	**Опис**
Тип програмного продукту	Навчально-прикладний мобільний застосунок
Призначення	Автоматизація базового моделювання та підготовки 3D-моделей до FDM-друку
Цільова аудиторія	Студенти, викладачі, початківці у сфері 3D-друку, користувачі навчальних FDM-принтерів
Основна технологія друку	FDM-друк
Основний сценарій роботи	Створення або імпорт моделі → налаштування параметрів → аналіз → перегляд → експорт або генерація G-code
Підтримувані примітиви	Куб, циліндр, сфера, конус, піраміда, тор
Підтримувані матеріали	PLA, ABS, PETG
Основні формати	STL, OBJ, 3MF, G-code
Мобільний клієнт	Flutter / Dart
Серверна частина	Python / FastAPI
База даних	SQLite
Архітектурний підхід	Клієнт-серверна архітектура
Основне обмеження	Застосунок не є промисловою CAD-системою або повноцінним професійним slicer-рішенням

Узагальнену схему роботи мобільного застосунку для підготовки 3D-моделі до FDM-друку наведено на рисунку 2.1.

Таким чином, розроблюваний програмний продукт є мобільним інструментом, який об’єднує базові функції моделювання та підготовки до FDM-друку в одному робочому сценарії. Його цінність полягає не у конкуренції з професійними CAD- або slicer-системами, а у спрощенні початкової взаємодії користувача з 3D-друком. Це визначає подальші вимоги до архітектури, структури даних, інтерфейсу та основних функціональних модулів системи.

Рис. 2.1 Узагальнена схема роботи мобільного застосунку для підготовки 3D-моделі до FDM-друку

2.2 Проєктування архітектури системи

Архітектура програмного продукту визначає загальну організацію системи, розподіл відповідальності між її компонентами та спосіб їх взаємодії. Для мобільного застосунку, який автоматизує процес моделювання та підготовки 3D-моделей до FDM-друку, архітектура має забезпечувати не лише відображення інтерфейсу користувача, а й обробку моделей, збереження даних, виконання розрахунків, аналіз друкованості, експорт файлів і генерацію G-code.

З огляду на функціональність розроблюваного програмного продукту доцільним є використання клієнт-серверної архітектури. У такій архітектурі мобільний клієнт відповідає за взаємодію з користувачем, а серверна частина — за обробку запитів, бізнес-логіку, роботу з базою даних і виконання операцій, які пов’язані з 3D-моделями. Такий підхід дозволяє розділити інтерфейсну логіку та логіку обробки даних, що робить систему більш структурованою і зручною для підтримки.

Мобільний клієнт реалізується з використанням Flutter та мови програмування Dart. Його основним завданням є надання користувачу зручного інтерфейсу для входу в систему, перегляду моделей, створення нових об’єктів, вибору шаблонів, імпорту файлів, налаштування параметрів друку, перегляду результатів аналізу та виконання експорту. Клієнтська частина не повинна містити всю складну логіку обробки моделей, оскільки це ускладнило б її підтримку та збільшило навантаження на мобільний пристрій.

Серверна частина реалізується з використанням Python та FastAPI. Вона виконує роль центрального рівня системи, який приймає запити від мобільного клієнта, перевіряє вхідні дані, взаємодіє з базою даних, виконує розрахунки та повертає клієнту результати. Через серверну частину проходять основні операції: авторизація користувача, отримання списку моделей, створення моделей із параметрів або шаблонів, збереження змін, імпорт STL/OBJ-файлів, аналіз друкованості, експорт у формати STL, OBJ, 3MF та генерація G-code.

Для збереження даних у системі використовується база даних SQLite. Вона підходить для навчально-прикладного MVP, оскільки є простою у використанні, не потребує окремого серверу баз даних і дозволяє зберігати основні сутності системи: користувачів, 3D-моделі, параметри друку, матеріали, профілі принтерів та службові дані. У межах бакалаврської роботи SQLite є достатнім рішенням для демонстрації логіки роботи застосунку та перевірки основних функцій.

Взаємодія між мобільним клієнтом і серверною частиною здійснюється через REST API. Клієнт надсилає HTTP-запити до відповідних маршрутів серверу, а сервер повертає дані у структурованому вигляді. Такий спосіб взаємодії є зручним для мобільних застосунків, оскільки дозволяє чітко розділити функції клієнтської і серверної частин. Наприклад, при створенні моделі користувач вводить параметри у мобільному застосунку, після чого клієнт надсилає запит на сервер, сервер обробляє ці дані, зберігає модель у базі даних і повертає оновлену інформацію для відображення.

У загальному вигляді архітектуру системи можна поділити на кілька логічних рівнів: рівень користувацького інтерфейсу, рівень API, рівень бізнес-логіки, рівень роботи з даними та рівень файлового експорту. Рівень користувацького інтерфейсу забезпечує взаємодію користувача із застосунком. Рівень API відповідає за приймання і передачу запитів. Рівень бізнес-логіки виконує операції над моделями, розрахунки, аналіз і підготовку результатів. Рівень даних забезпечує збереження інформації в SQLite. Рівень експорту формує файли у форматах STL, OBJ, 3MF та G-code.

На рисунку 2.2 показано загальну структуру системи, у якій користувач взаємодіє з мобільним клієнтом, а той через REST API звертається до серверної частини. Сервер обробляє запити, взаємодіє з базою даних, виконує розрахунки, аналізує моделі та формує результати у вигляді даних для відображення або файлів для експорту.

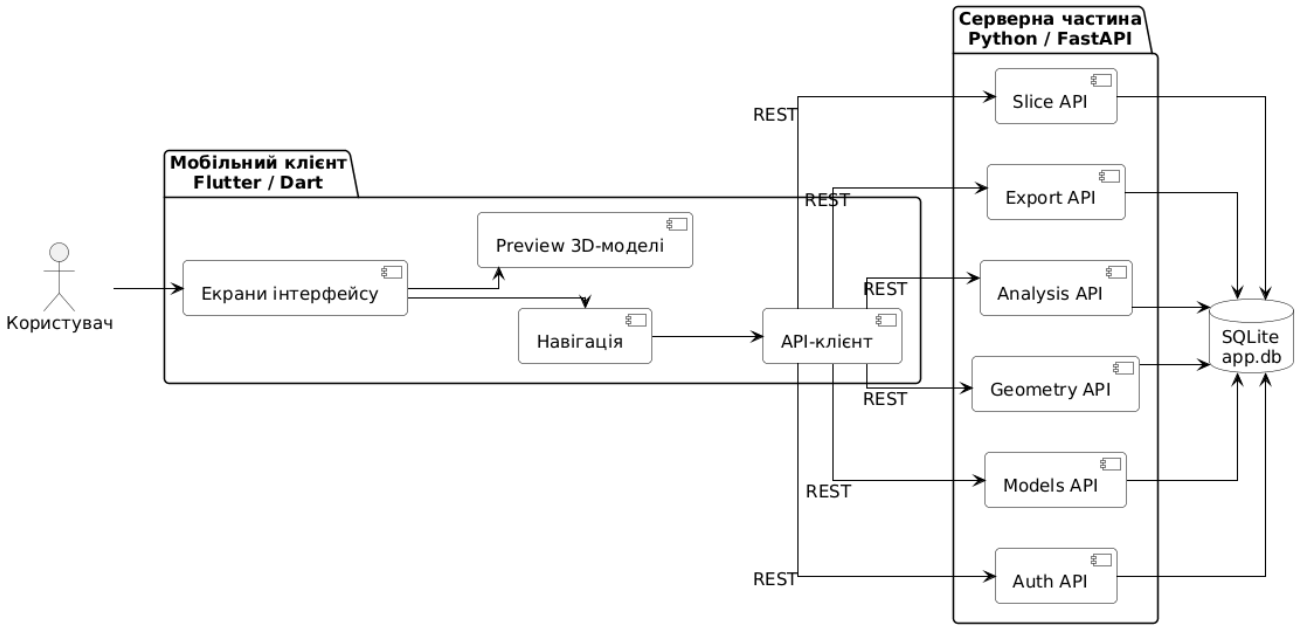


Рис. 2.2 Архітектура мобільного застосунку для автоматизації процесу моделювання під час 3D-друку

Основні компоненти архітектури системи наведено в таблиці 2.2.

Таблиця 2.2

Основні компоненти архітектури програмного продукту

Компонент системи	**Призначення**	**Основні функції**
Мобільний клієнт	Забезпечення взаємодії користувача із системою	Авторизація, введення параметрів, перегляд моделей, навігація, відображення результатів
REST API	Організація обміну даними між клієнтом і сервером	Приймання запитів, передача параметрів, повернення результатів
Модуль авторизації	Контроль доступу користувача до системи	Вхід користувача, перевірка облікових даних, отримання інформації про користувача
Модуль моделей	Робота з 3D-моделями	Створення, редагування, перегляд, видалення та збереження моделей
Модуль шаблонів	Створення моделей на основі готових заготовок	Надання списку шаблонів, створення моделі з шаблону
Модуль матеріалів і принтерів	Робота з технологічними параметрами FDM-друку	Вибір матеріалу, профілю принтера, температур, швидкості, заповнення
Модуль аналізу друкованості	Попередня оцінка моделі перед друком	Розрахунок об’єму, площі, маси, вартості, часу друку
Модуль імпорту	Завантаження готових моделей	Імпорт STL/OBJ-файлів
Модуль експорту	Формування вихідних файлів	Експорт STL, OBJ, 3MF та генерація G-code
База даних	Збереження даних системи	Зберігання користувачів, моделей, параметрів друку, матеріалів і профілів принтерів

Важливою перевагою такої архітектури є те, що мобільний клієнт залишається відносно простим і не перевантажується складними обчисленнями. Усі операції, пов’язані з обробкою моделей, розрахунками та формуванням файлів, виконуються на серверній частині. Це дозволяє централізувати логіку роботи з 3D-моделями та забезпечити однакову поведінку системи незалежно від мобільного пристрою користувача.

Клієнт-серверний підхід також спрощує подальший розвиток програмного продукту. У майбутньому серверну частину можна доповнити новими алгоритмами аналізу друкованості, підтримкою інших форматів, складнішими операціями над моделями або інтеграцією з зовнішніми сервісами. Мобільний клієнт при цьому може залишатися основним інтерфейсом користувача, а нові можливості будуть додаватися через розширення API та backend-логіки.

У межах проєктування також важливо визначити основні потоки даних у системі. Наприклад, під час створення параметричної моделі користувач вводить розміри та тип примітива в мобільному клієнті. Ці дані передаються на сервер через REST API. Сервер перевіряє параметри, створює запис моделі в базі даних і повертає клієнту результат. Під час аналізу друкованості клієнт надсилає запит для конкретної моделі, сервер обчислює необхідні показники та повертає їх для відображення. Під час експорту сервер формує відповідний файл і передає його користувачу.

Окремий потік даних пов’язаний з імпортом готових моделей. У цьому випадку користувач завантажує STL або OBJ-файл через мобільний клієнт. Сервер приймає файл, виконує базову обробку, зберігає інформацію про модель і робить її доступною для подальшого аналізу або експорту. Такий сценарій важливий, оскільки користувач може працювати не лише з моделями, створеними всередині застосунку, а й із зовнішніми файлами.

Ще один важливий сценарій стосується генерації G-code. Для цього система повинна мати дані про геометрію моделі, матеріал, параметри друку та профіль принтера. На основі цих даних серверна частина виконує slicing, формує послідовність команд і повертає користувачу файл G-code. Такий механізм

дозволяє поєднати моделювання та підготовку до друку в одному робочому процесі.

Таким чином, спроєктована архітектура відповідає задачам бакалаврської роботи та особливостям предметної галузі. Вона поєднує мобільний інтерфейс, серверну логіку, базу даних, модулі аналізу та експорту в єдину систему. Це дає змогу реалізувати основну ідею програмного продукту: надати користувачу простий, послідовний і зрозумілий інструмент для базового моделювання та підготовки 3D-моделей до FDM-друку.

2.3 Функціональні та нефункціональні вимоги до програмного застосунку

Вимоги є одним із ключових етапів проєктування програмного продукту, оскільки саме вимоги визначають, які функції повинна виконувати система, які обмеження необхідно врахувати під час реалізації та за якими критеріями можна буде перевірити готовий застосунок. Для мобільного застосунку, що автоматизує процес моделювання під час 3D-друку, вимоги мають охоплювати не лише стандартну роботу з користувачем і даними, а й специфічні операції предметної галузі: створення 3D-моделей, імпорт файлів, аналіз друкованості, налаштування параметрів FDM-друку, експорт моделей і генерацію G-code.

У межах цієї бакалаврської роботи вимоги до програмного забезпечення доцільно поділити на функціональні та нефункціональні. Функціональні вимоги описують конкретні можливості, які має надавати система користувачу. Нефункціональні вимоги визначають якісні характеристики роботи застосунку: швидкодію, стабільність, обмеження щодо файлів, підтримку форматів, коректність збереження даних і захищеність доступу. Такий поділ дозволяє не тільки спроєктувати систему, а й надалі перевірити її відповідність поставленим задачам під час тестування.

Під час формування вимог враховано призначення застосунку як навчально-прикладного MVP для студентів, початківців у сфері 3D-друку та користувачів навчальних FDM-принтерів. Тому вимоги не орієнтовані на промисловий рівень

професійних CAD-систем, а описують саме ті можливості, які потрібні для базового сценарію роботи: створити модель, налаштувати параметри, оцінити друкованість, переглянути результат і сформувати файл для подальшого використання.

Функціональні вимоги до розроблюваного мобільного застосунку наведено в таблиці 2.3.

Таблиця 2.3

Функціональні вимоги до мобільного застосунку

Код вимоги	**Функціональна вимога**	**Опис вимоги**	**Очікуваний результат**
FR-01	Авторизація користувача	Система повинна забезпечувати вхід користувача за логіном і паролем	Користувач отримує доступ до функцій застосунку після успішної авторизації
FR-02	Отримання даних поточного користувача	Система повинна надавати інформацію про авторизованого користувача	У застосунку відображається користувач і його роль
FR-03	Перегляд списку моделей	Користувач повинен мати змогу переглядати всі доступні йому 3D-моделі	На екрані відображається список моделей із базовою інформацією
FR-04	Створення параметричної моделі	Система повинна дозволяти створювати модель на основі базового примітива	У базі даних створюється нова модель із заданими параметрами
FR-05	Підтримка базових примітивів	Система повинна підтримувати створення куба, циліндра, сфери, конуса, піраміди та тора	Користувач може обрати потрібний тип геометричної форми
FR-06	Редагування параметрів моделі	Користувач повинен мати змогу змінювати розміри та основні параметри моделі	Змінені параметри зберігаються і використовуються під час аналізу та експорту
FR-07	Видалення моделі	Система повинна дозволяти видаляти непотрібну модель	Модель видаляється зі списку та бази даних
FR-08	Робота з шаблонами моделей	Система повинна надавати користувачу список готових шаблонів	Користувач може створити модель на основі шаблону
FR-09	Імпорт STL/OBJ-файлів	Система повинна підтримувати імпорт готових моделей у форматах STL та OBJ	Імпортована модель зберігається в системі та доступна для аналізу

Продовження таблиці 2.3

Функціональні вимоги до мобільного застосунку

FR-10	Вибір матеріалу друку	Користувач повинен мати змогу обрати матеріал PLA, ABS або PETG	Обраний матеріал використовується для розрахунків і параметрів друку
FR-11	Вибір профілю принтера	Система повинна підтримувати вибір профілю FDM-принтера	Параметри принтера враховуються під час аналізу та генерації G-code
FR-12	Налаштування параметрів друку	Користувач повинен мати змогу задавати висоту шару, заповнення, швидкість, температури, діаметр сопла та підтримки	Система зберігає набір параметрів для конкретної моделі
FR-13	Аналіз друкованості	Система повинна виконувати попередній аналіз моделі перед друком	Користувач отримує оцінку об’єму, маси, часу друку, вартості та інших показників
FR-14	Попередній перегляд моделі	Система повинна надавати візуальне представлення 3D-моделі	Користувач може оцінити форму моделі перед експортом
FR-15	Виконання операцій над моделлю	Система повинна підтримувати базові операції: масштабування, обертання, переміщення, дзеркальне відображення, taper, twist, skew та створення отвору	Користувач може змінювати модель без переходу в інше програмне середовище
FR-16	Експорт STL	Система повинна формувати файл моделі у форматі STL	Користувач отримує STL-файл для подальшого використання
FR-17	Експорт OBJ	Система повинна формувати файл моделі у форматі OBJ	Користувач отримує OBJ-файл
FR-18	Експорт 3MF	Система повинна формувати файл моделі у форматі 3MF	Користувач отримує 3MF-файл
FR-19	Генерація G-code	Система повинна формувати керуючі інструкції для FDM-принтера	Користувач отримує G-code-файл для друку
FR-20	Dashboard-модуль	Система повинна відображати загальну статистику щодо моделей і параметрів	Користувач бачить коротку аналітичну інформацію про роботу із застосунком

Функціональні вимоги показують, що застосунок повинен охоплювати повний базовий сценарій роботи з моделлю: від створення або імпорту до аналізу, експорту та генерації G-code. Важливо, що кожна функція має практичне призначення у предметній галузі. Наприклад, створення параметричної моделі відповідає етапу моделювання, вибір матеріалу та параметрів — етапу технологічної підготовки, аналіз друкованості — етапу попередньої перевірки, а експорт і G-code — етапу підготовки результату для подальшого використання у FDM-друці.

Окрім функціональних можливостей, важливо визначити нефункціональні вимоги. Вони повинні бути сформульовані так, щоб їх можна було перевірити під час тестування. Тому в таблиці 2.4 наведено не загальні побажання до системи, а конкретні вимоги з вимірюваними критеріями приймання.

Таблиця 2.4

Нефункціональні вимоги до мобільного застосунку

Код вимоги	**Нефункціональна вимога**	**Критерій приймання**
NFR-01	Час авторизації користувача	Авторизація має виконуватися не довше ніж за 2 секунди при стабільному з’єднанні із сервером
NFR-02	Час завантаження списку моделей	Список до 50 моделей має завантажуватися не довше ніж за 3 секунди
NFR-03	Час створення параметричної моделі	Створення моделі з базового примітива має виконуватися не довше ніж за 3 секунди
NFR-04	Час збереження змінених параметрів	Збереження змін розмірів, матеріалу або параметрів друку має виконуватися не довше ніж за 2 секунди
NFR-05	Час формування попереднього перегляду	2D, ізометричний або 3D-перегляд моделі до 10 000 трикутників має формуватися не довше ніж за 5 секунд
NFR-06	Час виконання аналізу друкованості	Аналіз типової навчальної моделі має виконуватися не довше ніж за 5 секунд
NFR-07	Час експорту моделі	Експорт STL, OBJ або 3MF для моделі розміром до 20 МБ має виконуватися не довше ніж за 10 секунд
NFR-08	Час генерації G-code	Генерація G-code для простої моделі розміром до 100×100×100 мм має виконуватися не довше ніж за 15 секунд
NFR-09	Підтримка форматів	Система повинна підтримувати імпорт щонайменше 2 форматів: STL і OBJ, та експорт щонайменше 4 форматів: STL, OBJ, 3MF і G-code

Продовження таблиці 2.4

Нефункціональні вимоги до мобільного застосунку

Код вимоги	Нефункціональна вимога	Критерій приймання
NFR-10	Обмеження розміру імпортованого файлу	Система повинна коректно обробляти STL/OBJ-файли розміром до 20 МБ
NFR-11	Точність розрахунків	Похибка розрахунку маси матеріалу та орієнтовної вартості не повинна перевищувати 5% відносно заданих параметрів матеріалу та моделі
NFR-12	Стабільність роботи	Після виконання 30 послідовних операцій створення, редагування, аналізу та експорту застосунок не повинен аварійно завершувати роботу
NFR-13	Захист доступу	Запити до захищених функцій без токена авторизації мають відхилятися у 100% випадків
NFR-14	Коректність збереження даних	Після успішного створення або редагування моделі дані мають зберігатися у базі даних у 100% успішних операцій
NFR-15	Сумісність інтерфейсу	Інтерфейс має коректно відображатися на екранах шириною від 360 px до 1080 px
NFR-16	Зручність базового сценарію	Користувач повинен мати змогу створити модель із шаблону не більше ніж за 5 дій після входу в систему
NFR-17	Обробка помилок	При помилці імпорту, експорту або аналізу система має показати повідомлення користувачу не пізніше ніж за 2 секунди після отримання відповіді від сервера

Наведені нефункціональні вимоги є важливими для подальшого тестування, оскільки дозволяють оцінити не тільки наявність функцій, а й якість їх виконання. Наприклад, функція генерації G-code сама по собі не є достатньою, якщо вона виконується занадто довго або нестабільно. Так само імпорт STL/OBJ-файлів повинен не лише існувати, а й мати чітке обмеження за розміром файлу, щоб результат можна було перевірити під час тестування.

Для моделювання вимог також доцільно визначити ролі користувачів системи. У межах розроблюваного застосунку передбачено дві основні ролі: студент і адміністратор. Роль студента відповідає звичайному користувачу, який створює моделі, працює з шаблонами, налаштовує параметри друку, виконує аналіз

і експортує файли. Роль адміністратора може використовуватися для розширеного доступу до даних, перевірки системи, керування демонстраційними даними або роботи з матеріалами та профілями принтерів.

Основні ролі користувачів системи наведено в таблиці 2.5.

Таблиця 2.5

Ролі користувачів мобільного застосунку

Роль користувача	**Призначення ролі**	**Основні можливості**
Студент	Основний користувач застосунку	Авторизація, створення моделей, використання шаблонів, імпорт, аналіз, перегляд, експорт, генерація G-code
Адміністратор	Користувач із розширеними можливостями для контролю та демонстрації системи	Перегляд даних, робота з моделями, матеріалами, профілями принтерів і тестовими сценаріями

Загальна логіка вимог до системи базується на тому, що користувач повинен мати змогу виконати основні етапи підготовки до FDM-друку без переходу між кількома окремими програмними середовищами. Тому вимоги охоплюють не тільки інтерфейсні функції, а й серверну логіку, обробку файлів, розрахунки, збереження даних та експорт результатів.

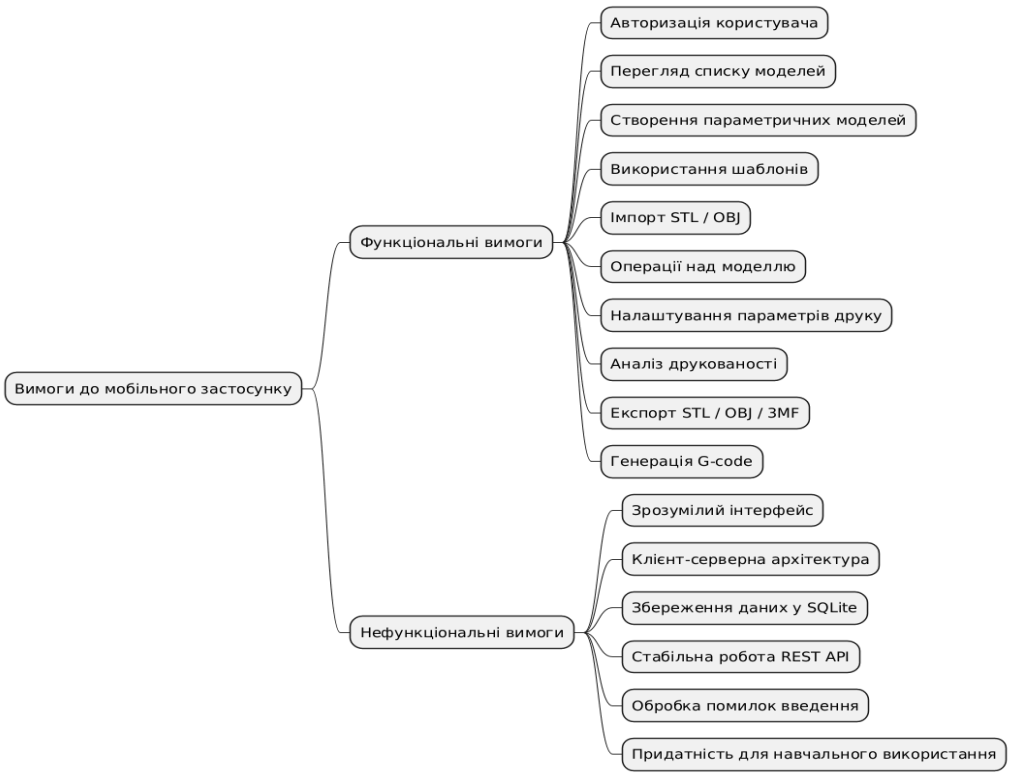


Рис. 2.3 Узагальнена модель вимог до мобільного застосунку

На рисунку 2.3 показано поділ вимог на три взаємопов’язані групи: функціональні вимоги, нефункціональні вимоги та ролі користувачів. У сукупності вони визначають, які можливості має забезпечувати застосунок, з якою якістю вони повинні виконуватися та які сценарії доступні різним користувачам системи.

Отже, змодельовані вимоги визначають основу для подальшого проєктування системи. Функціональні вимоги показують, які можливості має надати застосунок, нефункціональні вимоги встановлюють критерії якості та перевірки, а ролі користувачів визначають сценарії доступу до системи. На основі цих вимог у наступних підрозділах доцільно перейти до проєктування сценаріїв використання, структури бази даних та інтерфейсу користувача.

2.4 Проєктування сценаріїв використання системи

Проєктування сценаріїв використання необхідне для того, щоб визначити, яким чином користувач буде взаємодіяти з мобільним застосунком під час виконання основних задач. На відміну від загального переліку функціональних вимог, сценарії використання описують послідовність дій користувача та реакцію системи на ці дії. Для розроблюваного застосунку це особливо важливо, оскільки його основна мета полягає у спрощенні процесу моделювання та підготовки 3D-моделей до FDM-друку.

У системі передбачено дві основні ролі користувачів: студент і адміністратор. Студент є основним користувачем застосунку. Він може авторизуватися, переглядати список моделей, створювати нові параметричні моделі, використовувати шаблони, імпортувати STL/OBJ-файли, налаштовувати параметри друку, запускати аналіз друкованості, переглядати модель, експортувати її у потрібному форматі та генерувати G-code. Адміністратор має розширені можливості для контролю даних, роботи з матеріалами, профілями принтерів, демонстраційними моделями та перевірки працездатності системи.

Основна логіка використання застосунку побудована навколо послідовного проходження користувачем етапів підготовки моделі до друку. Після входу в систему користувач переходить до списку моделей або dashboard-модуля. Далі він може створити модель з параметричного примітива, скористатися готовим шаблоном або імпортувати вже існуючий файл. Після цього модель може бути переглянута, налаштована, проаналізована та експортована у форматах STL, OBJ, 3MF або G-code.

Основні сценарії використання розроблюваного застосунку наведено в таблиці 2.6.

Таблиця 2.6

Основні сценарії використання мобільного застосунку

Код сценарію	**Назва сценарію**	**Учасник**	**Очікуваний результат**
UC-01	Авторизація користувача	Студент, адміністратор	Користувач отримує доступ до функцій системи
UC-02	Перегляд списку моделей	Студент, адміністратор	На екрані відображається перелік доступних моделей
UC-03	Створення параметричної моделі	Студент	У системі створюється нова модель із заданими параметрами
UC-04	Створення моделі з шаблону	Студент	Користувач отримує модель на основі готової заготовки
UC-05	Імпорт STL/OBJ-файлу	Студент	Імпортована модель зберігається в системі
UC-06	Налаштування параметрів друку	Студент	Для моделі зберігаються параметри друку
UC-07	Аналіз друкованості	Студент	Користувач отримує результати розрахунків
UC-08	Перегляд моделі	Студент	Користувач візуально оцінює форму та параметри моделі
UC-09	Експорт STL/OBJ/3MF	Студент	Користувач отримує файл моделі в обраному форматі
UC-10	Генерація G-code	Студент	Користувач отримує файл керуючих інструкцій для FDM-принтера
UC-11	Робота з dashboard	Студент, адміністратор	Користувач бачить узагальнену інформацію про моделі та параметри
UC-12	Керування довідковими даними	Адміністратор	Адміністратор працює з матеріалами, профілями принтерів і демонстраційними даними

Одним із ключових сценаріїв є створення параметричної 3D-моделі. Він починається після авторизації користувача. Користувач переходить на екран створення моделі, обирає тип геометричного примітива, наприклад куб, циліндр, сферу, конус, піраміду або тор, задає розміри та підтверджує створення. Мобільний клієнт передає введені параметри на серверну частину через REST API. Сервер перевіряє коректність даних, створює запис моделі в базі даних і повертає клієнту результат. Після цього модель з'являється у списку та стає доступною для перегляду, аналізу, редагування параметрів і експорту.

2.5 Проектування діаграм послідовності

Діаграма послідовності створення параметричної 3D-моделі представлена на рисунку 2.5.

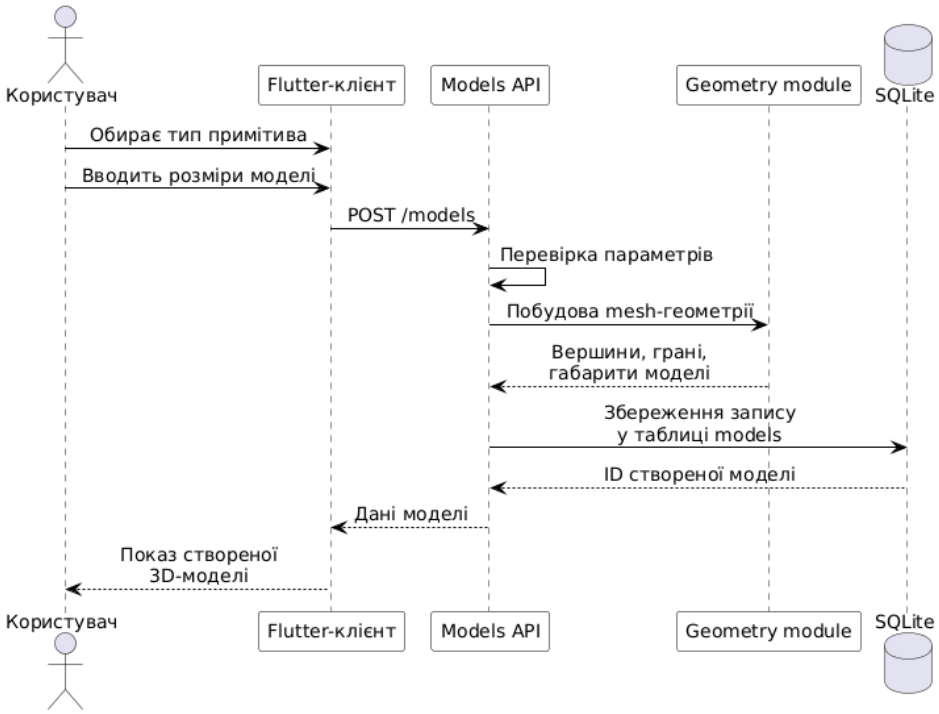


Рис. 2.5 Діаграма послідовності створення параметричної 3D-моделі

Іншим важливим сценарієм є створення моделі з шаблону. У цьому випадку користувач не вводить усі параметри з нуля, а обирає готову заготовку із

запропонованого списку. Такий сценарій потрібний для навчальних і типових прикладних задач, коли користувачу необхідно швидко створити калібрувальний куб, простий корпус, технічну форму або інший базовий об’єкт. Після вибору шаблону сервер створює нову модель на його основі та додає її до списку моделей користувача. Це скорочує кількість дій і робить роботу із застосунком простішою.

Сценарій імпорту STL/OBJ-файлу передбачає роботу з уже готовими моделями, створеними в інших програмних середовищах. Користувач обирає файл, після чого мобільний клієнт передає його на сервер. Сервер виконує базову обробку файлу, зберігає інформацію про модель і робить її доступною для подальшого перегляду, аналізу або експорту. Цей сценарій є важливим, оскільки розширює можливості застосунку і дозволяє працювати не лише з моделями, створеними всередині системи.

Окрему роль відіграє сценарій налаштування параметрів друку. Користувач обирає матеріал, профіль принтера, висоту шару, відсоток заповнення, швидкість друку, температуру сопла, температуру друкованого столу, діаметр сопла та потребу в підтримках. Ці параметри зберігаються для конкретної моделі та використовуються під час аналізу друкованості й генерації G-code. Завдяки цьому користувач бачить не просто геометричну модель, а модель у контексті реального процесу FDM-друку.

Одним із найважливіших сценаріїв є аналіз друкованості моделі. Після вибору моделі користувач запускає аналіз, а система передає відповідний запит на сервер. Серверна частина використовує дані про геометрію моделі, матеріал, параметри друку та профіль принтера, після чого розраховує основні показники: об’єм, площу поверхні, масу матеріалу, довжину філаменту, орієнтовний час друку та вартість. Результати повертаються на мобільний клієнт і відображаються користувачу у зрозумілому вигляді.

Діаграма послідовності аналізу друкованості моделі представлена на рисунку 2.6.

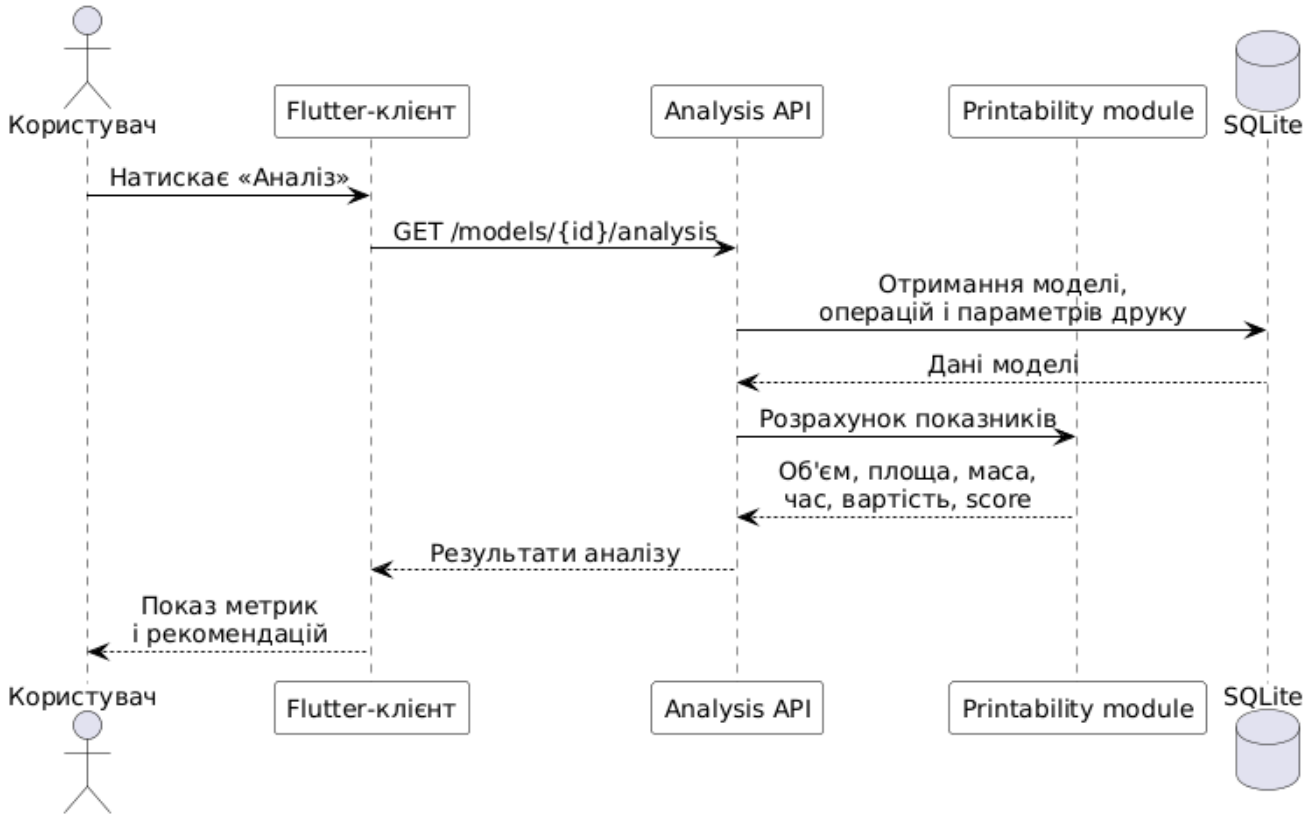


Рис. 2.6 Діаграма послідовності аналізу друкованості моделі

Завершальним сценарієм є експорт моделі або генерація G-code. Після створення, налаштування та аналізу користувач обирає потрібний формат результату. Якщо обрано STL, OBJ або 3MF, система формує файл моделі для подальшого використання в інших програмах. Якщо користувач обирає G-code, серверна частина виконує підготовку керуючих інструкцій для FDM-принтера з урахуванням параметрів друку. У результаті користувач отримує файл, який може бути використаний для подальшого друку або демонстрації роботи системи.

Діаграма послідовності експорту моделі та генерації G-code представлена на рисунку 2.7.

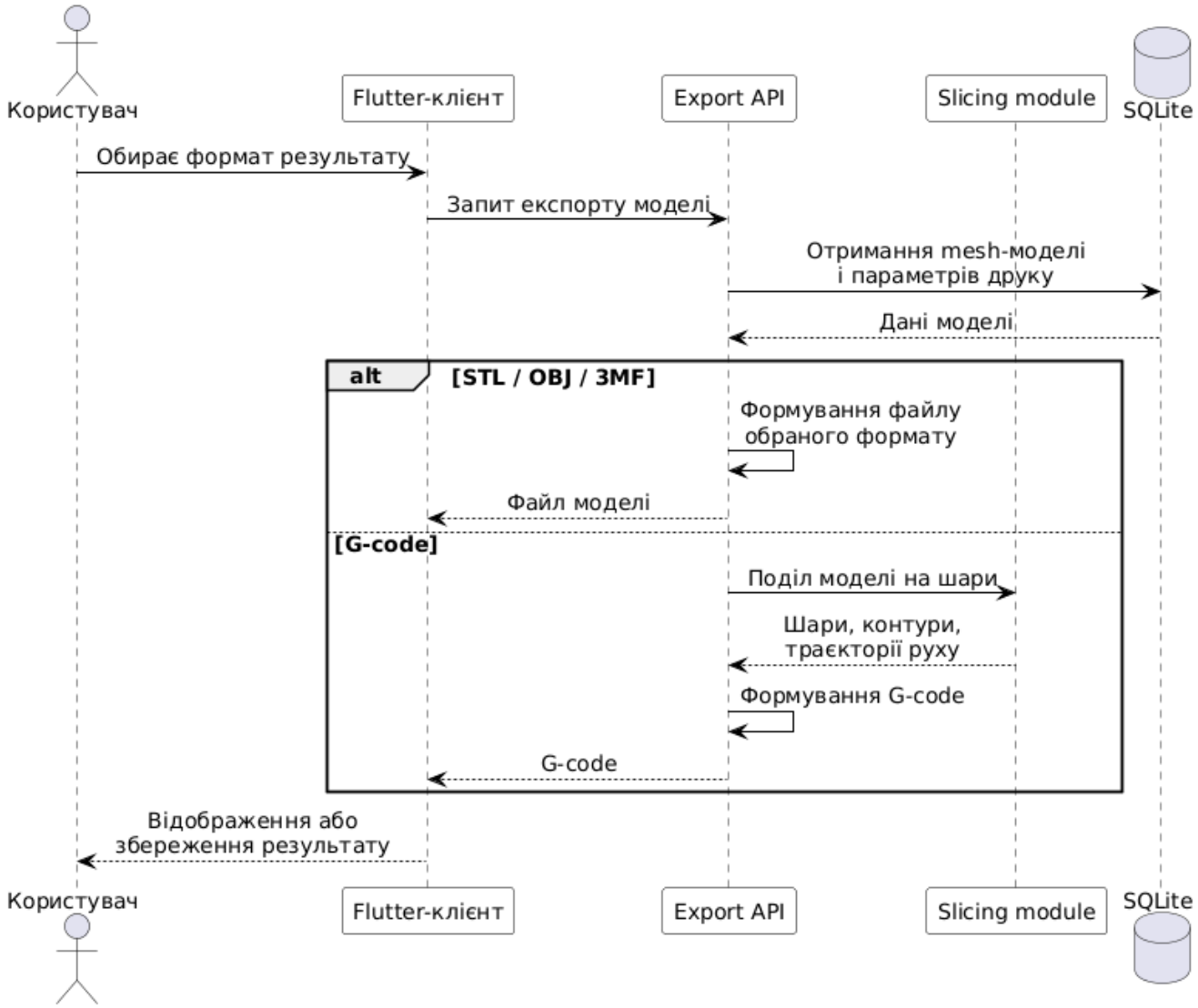


Рис. 2.7 Діаграма послідовності експорту моделі та генерації G-code

Під час проєктування сценаріїв важливо врахувати не тільки успішне виконання операцій, а й можливі помилкові ситуації. Наприклад, користувач може ввести некоректні розміри моделі, завантажити файл непідтримуваного формату, спробувати виконати аналіз без достатніх параметрів або запустити експорт для пошкодженої моделі. У таких випадках система повинна не просто припинити виконання операції, а показати зрозуміле повідомлення про причину помилки. Це важливо для цільової аудиторії, оскільки студенти й початківці можуть не мати достатнього досвіду для самостійного визначення проблеми.

Узагальнений сценарій роботи користувача із застосунком можна подати як послідовність: авторизація → вибір або створення моделі → налаштування

параметрів друку → перегляд моделі → аналіз друкованості → експорт або генерація G-code. Така структура відповідає логіці підготовки 3D-моделі до FDM-друку та дозволяє користувачу виконати основні дії в одному середовищі.

Діаграма узагальненого сценарію роботи користувача із застосунком представлена на рисунку 2.8.

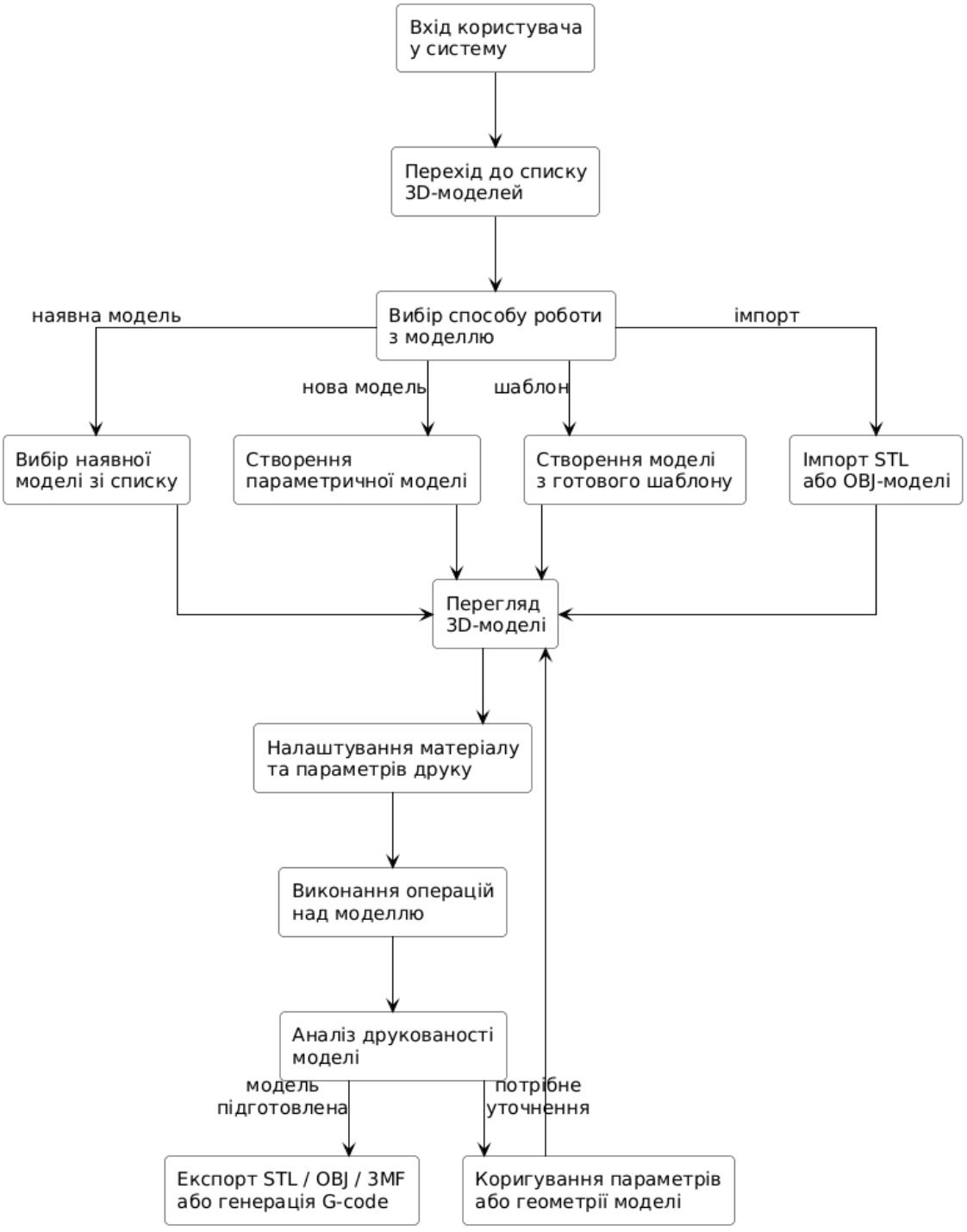


Рис. 2.8 Узагальнений сценарій роботи користувача із застосунком

Таким чином, проєктування сценаріїв використання дозволяє описати роботу системи з позиції користувача. Для розроблюваного мобільного застосунку основними є сценарії авторизації, створення параметричної моделі, створення моделі з шаблону, імпорту STL/OBJ, налаштування параметрів друку, аналізу друкованості, перегляду, експорту та генерації G-code. Їх реалізація забезпечує послідовний і зрозумілий шлях підготовки 3D-моделі до FDM-друку, що відповідає меті бакалаврської роботи.

2.5 Проєктування структури бази даних

Проєктування структури бази даних є важливим етапом створення мобільного застосунку, оскільки саме база даних забезпечує збереження основної інформації, необхідної для роботи системи. У розроблюваному програмному продукті база даних використовується для зберігання користувачів, створених 3D-моделей, параметрів друку, матеріалів, профілів принтерів, шаблонів моделей та службових даних, які використовуються під час аналізу, імпорту, експорту й генерації G-code.

Для реалізації програмного продукту використовується SQLite. Такий вибір є доцільним для навчально-прикладного MVP, оскільки SQLite не потребує окремого серверу баз даних, є простою в налаштуванні та достатньою для збереження структурованих даних у межах бакалаврського проєкту. У контексті розроблюваного застосунку SQLite дозволяє зберігати всі основні сутності системи та забезпечує швидкий доступ до них під час виконання запитів серверною частиною.

Структура бази даних повинна відповідати основним сценаріям роботи користувача. Після авторизації користувач отримує доступ до власних моделей, може створювати нові об’єкти, редагувати їх параметри, запускати аналіз друкованості, обирати матеріал і профіль принтера, виконувати експорт або генерацію G-code. Отже, база даних має підтримувати зв’язок між користувачем, моделями, параметрами друку та допоміжними довідковими сутностями.

Однією з основних сутностей системи є користувач. Вона потрібна для організації авторизації, розмежування доступу та збереження інформації про те, кому належать створені моделі. У системі передбачено ролі користувачів, зокрема студент і адміністратор. Студент працює з моделями, параметрами друку, аналізом і експортом, а адміністратор може мати розширені можливості для контролю демонстраційних даних, матеріалів і профілів принтерів. Збереження ролі користувача дозволяє надалі розширювати систему та додавати різні рівні доступу.

Центральною сутністю бази даних є 3D-модель. Вона описує об’єкт, з яким працює користувач. Для параметричних моделей зберігається тип геометричного примітива, назва моделі, розміри, дата створення, дата оновлення та зв’язок із користувачем. До типів примітивів належать куб, циліндр, сфера, конус, піраміда та тор. Для імпортованих моделей додатково може зберігатися інформація про вихідний файл, формат імпорту та дані сітки, якщо модель була завантажена у форматі STL або OBJ.

Окремо від основної інформації про модель доцільно зберігати параметри друку. Це пов’язано з тим, що одна й та сама модель може мати різні технологічні налаштування залежно від матеріалу, принтера, висоти шару, заповнення або температури. Параметри друку включають матеріал, висоту шару, відсоток заповнення, швидкість друку, температуру сопла, температуру друкованого столу, діаметр сопла, потребу в підтримках, орієнтовну ціну філаменту та інші характеристики, необхідні для аналізу друкованості й генерації G-code.

Сутність матеріалу використовується як довідкова. Вона містить інформацію про тип філаменту, наприклад PLA, ABS або PETG, а також його базові властивості, які можуть застосовуватися під час розрахунків. До таких властивостей належать густина, рекомендована температура сопла, температура друкованого столу та орієнтовна вартість матеріалу. Зберігання матеріалів у вигляді окремої сутності є зручним, оскільки дозволяє не дублювати ці дані в кожній моделі та спрощує подальше розширення списку підтримуваних матеріалів.

Сутність профілю принтера також виконує довідкову роль. Вона описує технічні можливості FDM-принтера, зокрема розміри робочої області, діаметр

сопла, стандартні температурні параметри та інші характеристики. Наявність профілів принтерів дозволяє враховувати обмеження конкретного обладнання під час аналізу моделі. Наприклад, якщо габарити моделі перевищують робочу область принтера, система може визначити таку модель як проблемну для друку на вибраному обладнанні.

Для спрощення створення моделей у системі передбачено сутність шаблону. Шаблон зберігає готову конфігурацію моделі, яку користувач може використати як основу для нового об’єкта. Це може бути калібрувальний куб, корпус, технічна форма або інший типовий об’єкт. Такий підхід скорочує кількість дій користувача та робить застосунок зручнішим для навчальних сценаріїв. Після вибору шаблону система створює нову модель на його основі, а сам шаблон залишається незмінним.

Окремий блок даних пов’язаний з операціями над моделями. У застосунку передбачені операції масштабування, обертання, переміщення, дзеркального відображення, деформації taper, twist, skew та створення отвору. Збереження інформації про операції дозволяє відображати історію змін моделі або застосовувати набір перетворень під час формування геометрії. Для кожної операції доцільно зберігати тип операції, параметри, дату виконання та зв’язок із відповідною моделлю.

Для імпортованих моделей може використовуватися окрема сутність або окремий набір полів, пов’язаний із mesh-даними. Такий підхід потрібний, оскільки модель, створена параметрично, і модель, імпортована з STL/OBJ-файлу, мають різну природу. Параметрична модель описується числовими параметрами, а імпортована модель може містити сітку з вершин і граней. Тому в базі даних необхідно передбачити можливість збереження інформації про джерело моделі та її геометричне представлення.

Основні зв’язки між сутностями будуються навколо користувача та моделі. Один користувач може мати багато моделей. Кожна модель належить конкретному користувачу. Одна модель може мати один набір актуальних параметрів друку, а також пов’язуватися з матеріалом і профілем принтера. Модель може бути

створена вручну, на основі шаблону або через імпорт файлу. Також модель може мати набір операцій, які були виконані над нею під час редагування.

На рисунку 2.9 подано логічну структуру бази даних мобільного застосунку.

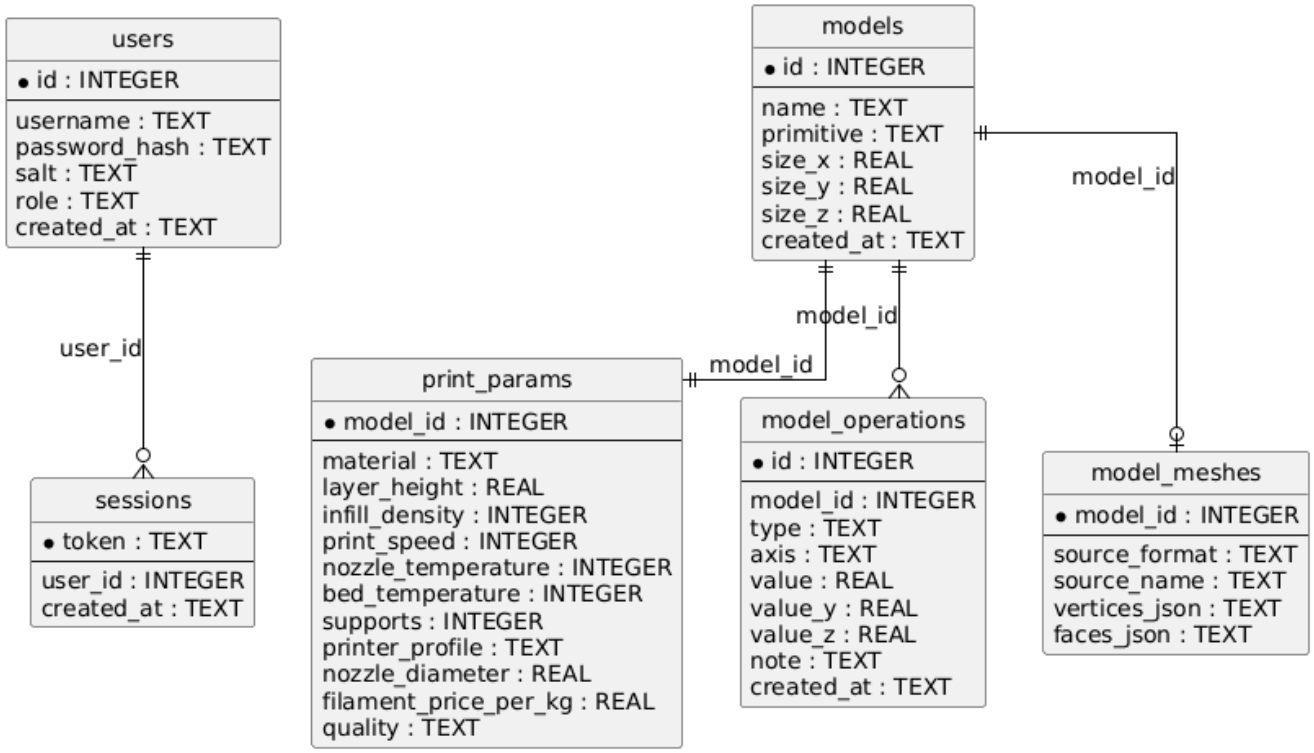


Рис. 2.9 Структура бази даних мобільного застосунку

Центральною сутністю є Models, оскільки саме модель об’єднує більшість функцій системи: створення параметричної форми, імпорт, налаштування параметрів друку, аналіз, операції над геометрією та експорт. Зв’язок Users з Models показує належність моделей конкретним користувачам, а зв’язки з PrintParams, Materials і Printers забезпечують підготовку моделі до FDM-друку.

Проєктована структура бази даних повинна забезпечувати цілісність інформації. Наприклад, модель не повинна існувати без користувача, якому вона належить. Параметри друку повинні бути пов’язані з конкретною моделлю. Матеріал і профіль принтера мають використовуватися під час аналізу друкованості та генерації G-code. Такий підхід дозволяє уникнути ситуацій, коли модель існує без необхідних даних для подальшої обробки.

Важливо також врахувати подальше розширення системи. У майбутньому до бази даних можна додати підтримку більшої кількості матеріалів, нових типів принтерів, історії експорту, результатів тестових друків або збереження декількох варіантів параметрів для однієї моделі. Тому структура бази даних має бути достатньо простою для поточної реалізації, але не повинна обмежувати можливість розвитку програмного продукту.

У межах цієї бакалаврської роботи база даних виконує не лише роль сховища даних, а й підтримує основну логіку застосунку. Через неї забезпечується зв’язок між користувачем, моделлю, параметрами друку, матеріалом, принтером і результатами обробки. Саме тому правильне проєктування структури бази даних є необхідною умовою стабільної роботи системи.

Таким чином, структура бази даних розроблюваного мобільного застосунку орієнтована на підтримку повного базового сценарію підготовки 3D-моделі до FDM-друку. Вона дозволяє зберігати користувачів, моделі, параметри друку, довідкові дані, шаблони та інформацію про операції над моделями. Це створює основу для реалізації функцій створення, аналізу, перегляду, експорту та генерації G-code в межах єдиного програмного продукту.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Проєктування інтерфейсу користувача

Проєктування інтерфейсу користувача є важливим етапом розробки мобільного застосунку, оскільки саме через інтерфейс користувач взаємодіє з основними функціями системи. У випадку застосунку для автоматизації процесу моделювання під час 3D-друку інтерфейс має не лише відображати дані, а й допомагати користувачу послідовно пройти основні етапи роботи з моделлю: авторизацію, вибір або створення моделі, налаштування параметрів друку, перегляд, аналіз друкованості, експорт і генерацію G-code.

Оскільки програмний продукт орієнтований на студентів, початківців у сфері 3D-друку та користувачів навчальних FDM-принтерів, інтерфейс не повинен бути перевантажений складними професійними інструментами. Його головне завдання полягає у тому, щоб зробити роботу з моделлю зрозумілою і послідовною. Користувач має бачити, на якому етапі підготовки він перебуває, які дії доступні, які параметри потрібно заповнити та який результат буде отримано після виконання операції.

Під час проєктування інтерфейсу важливо враховувати мобільний формат застосунку. На відміну від десктопних CAD-систем, мобільний екран має обмежений простір, тому елементи керування повинні бути згруповані за змістом. Основні дії мають бути винесені на окремі екрани або зрозумілі блоки, щоб користувач не втрачався у великій кількості параметрів. Саме тому інтерфейс доцільно будувати не як складну панель професійного редактора, а як набір послідовних екранів, кожен із яких відповідає певному етапу підготовки моделі до друку.

Першим екраном застосунку є екран авторизації. Його призначення полягає у забезпеченні доступу користувача до системи. На цьому екрані користувач вводить логін і пароль, після чого система перевіряє облікові дані та відкриває доступ до основних функцій. Для навчально-прикладного застосунку важливо, щоб

екран входу був простим і не містив зайвих елементів, оскільки його основна функція — швидко перевести користувача до роботи з моделями.

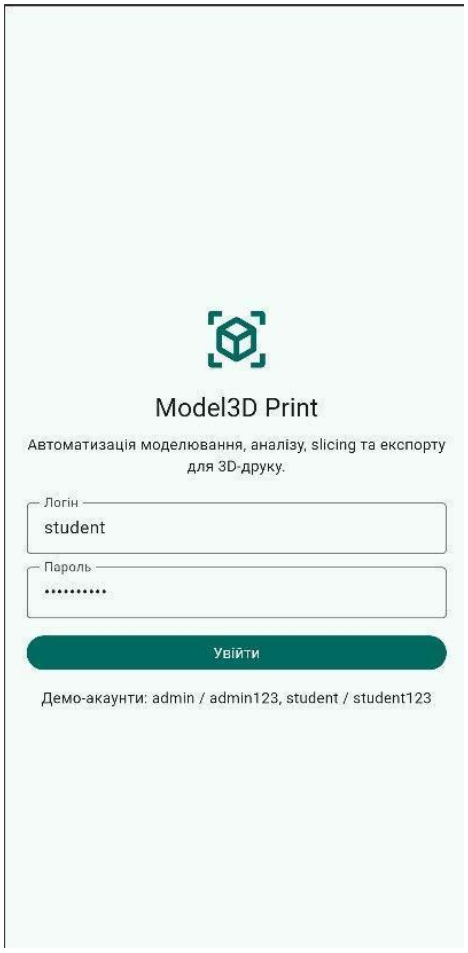


Рис. 3.1 Екран авторизації користувача в мобільному застосунку

Після авторизації користувач переходить до основної частини застосунку. Одним із центральних екранів є список моделей. На ньому мають відображатися моделі, доступні користувачу, їх назви, типи, основні параметри та можливість перейти до детальної інформації. Такий екран виконує роль стартової точки для подальшої роботи: користувач може відкрити вже створену модель, створити нову, скористатися шаблоном або імпортувати файл. Це відповідає логіці застосунку, де всі дії починаються з вибору або створення конкретної моделі.

Окремим елементом головного інтерфейсу є dashboard-модуль. Його призначення полягає у відображенні узагальненої інформації про моделі, параметри друку та стан роботи користувача із застосунком. Dashboard не повинен замінювати детальний аналіз моделі, але може допомогти швидко оцінити кількість створених моделей, використані матеріали, доступні шаблони або інші короткі

показники. Такий екран робить застосунок більш інформативним і зручним для навчального використання.

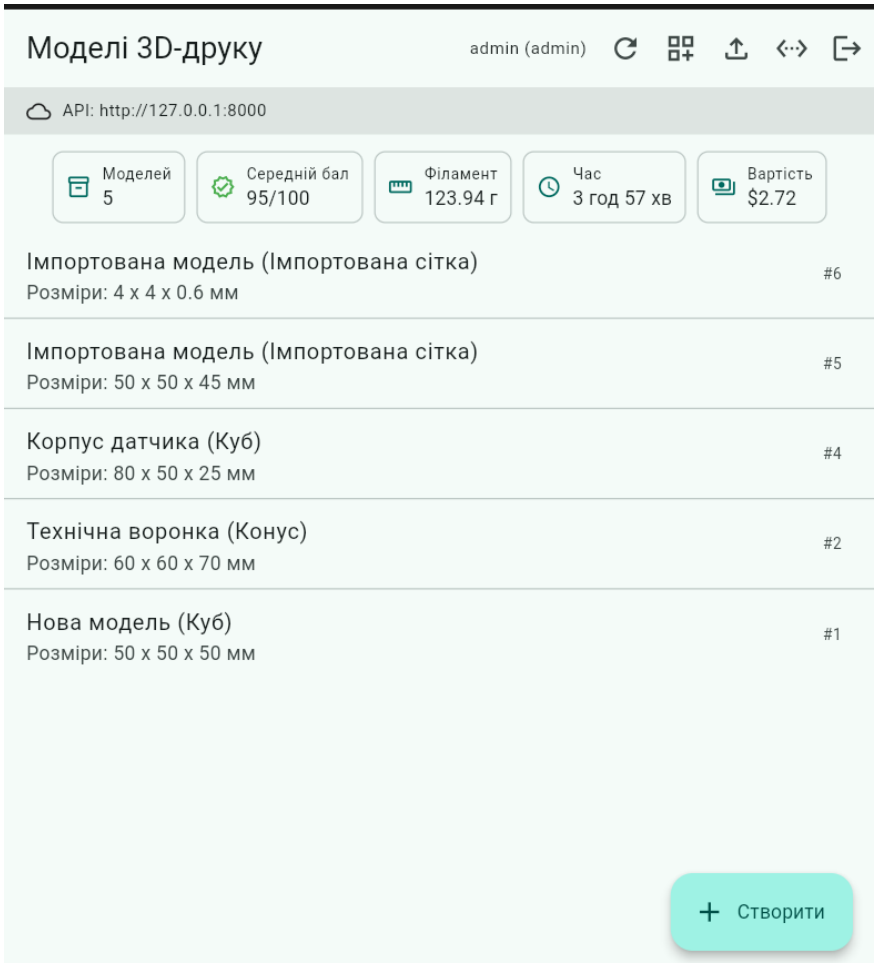


Рис. 3.2 Екран списку моделей та dashboard-модуля

Екран створення моделі має бути побудований так, щоб користувач міг швидко задати основні параметри майбутнього об’єкта. На цьому екрані доцільно передбачити вибір типу геометричного примітива: куба, циліндра, сфери, конуса, піраміди або тора. Після вибору типу моделі користувач вводить числові параметри, наприклад ширину, висоту, глибину, радіус або інші характеристики, які потрібні для формування геометрії. Такий підхід дозволяє уникнути складного ручного моделювання й водночас дає користувачу контроль над основними розмірами об’єкта.

Для прискорення роботи доцільно передбачити екран шаблонів. Він має містити перелік готових заготовок, які користувач може використати як основу для нової моделі. Наприклад, це можуть бути навчальні або типові об’єкти: калібрувальний куб, технічна форма, корпус або інша проста модель.

Використання шаблонів є важливим для цільової аудиторії застосунку, оскільки дозволяє скоротити кількість дій і швидше перейти до аналізу та підготовки до друку.

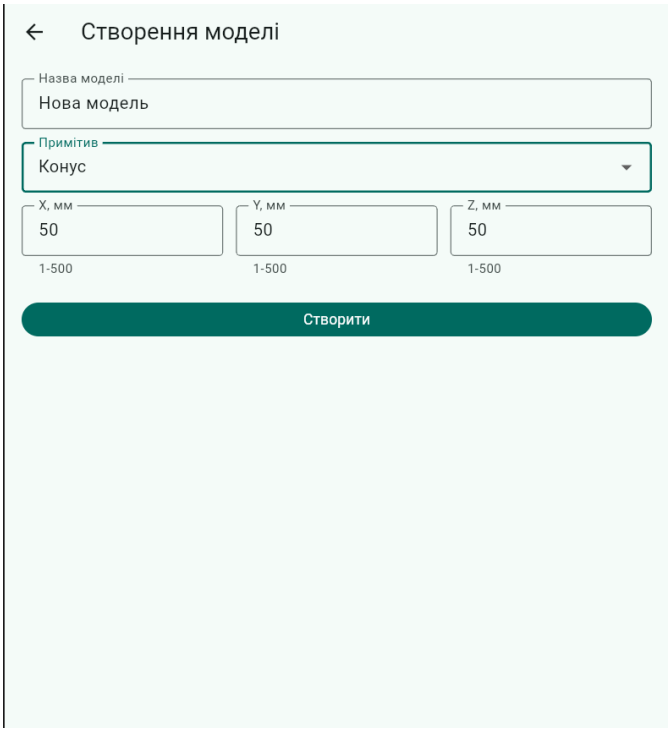


Рис. 3.3 Екран створення параметричної моделі та вибору шаблону

Екран деталей моделі є одним із найважливіших у застосунку. На ньому користувач повинен бачити основну інформацію про вибрану модель: назву, тип, розміри, матеріал, параметри друку, дату створення та доступні дії. Саме з цього екрана користувач може перейти до перегляду моделі, редагування параметрів, запуску аналізу друкованості, виконання операцій над моделлю або експорту. Тому екран деталей має бути організований так, щоб не перевантажувати користувача, але водночас забезпечити швидкий доступ до ключових функцій.

Для роботи з параметрами друку доцільно використовувати окремий екран або окремий блок у деталях моделі. У ньому користувач має змогу обрати матеріал, профіль принтера, висоту шару, відсоток заповнення, швидкість друку, температуру сопла, температуру друкованого столу, діаметр сопла та потребу в підтримках. Оскільки ці параметри суттєво впливають на результат FDM-друку, інтерфейс повинен подавати їх у зрозумілому вигляді, без надмірної кількості складних технічних налаштувань.

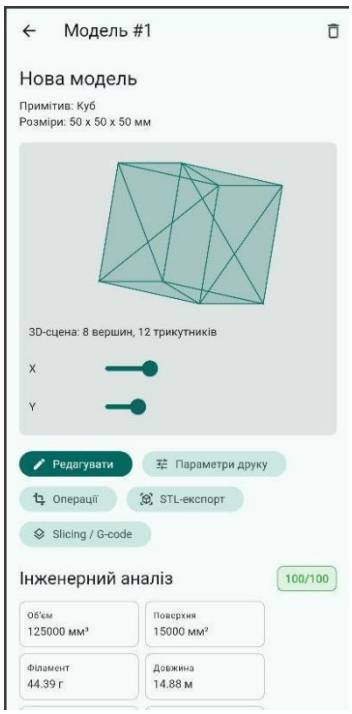


Рис. 3.4 Екран деталей моделі та налаштування параметрів друку

Окрему роль відіграє екран попереднього перегляду моделі. Його призначення полягає у візуальній перевірці форми об’єкта. Користувач повинен мати можливість побачити модель до експорту або генерації G-code, щоб переконатися, що вона відповідає очікуваному результату. Для навчального застосунку достатньо передбачити спрощений 2D або ізометричний перегляд, а також 3D-представлення моделі. Це дозволяє поєднати числові параметри з візуальним результатом, що важливо для розуміння процесу моделювання.

Екран аналізу друкованості має відображати результати розрахунків, які система виконує на основі геометрії моделі, матеріалу, профілю принтера та параметрів друку. До таких результатів належать об’єм, площа поверхні, маса матеріалу, довжина філаменту, орієнтовний час друку та приблизна вартість. Дані мають бути подані у вигляді зрозумілих показників, щоб користувач міг швидко оцінити модель перед друком. Якщо система виявляє потенційні проблеми, вони повинні бути показані окремо у вигляді повідомлень або попереджень.

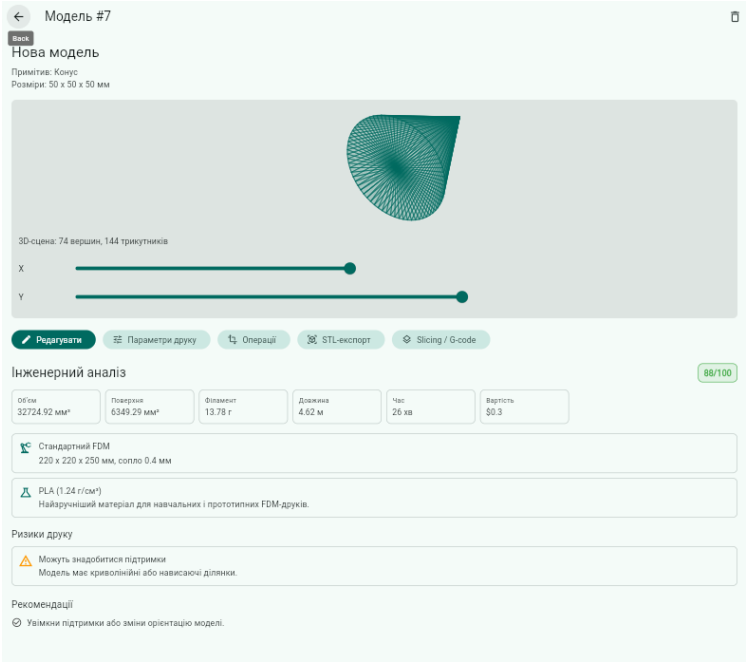


Рис. 3.5 Екран попереднього перегляду та аналізу друкованості моделі

Завершальним етапом роботи користувача є експорт моделі або генерація G-code. Для цього в інтерфейсі має бути передбачений екран або блок експорту, де користувач обирає потрібний формат результату: STL, OBJ, 3MF або G-code. Якщо користувач обирає формат моделі, система формує відповідний файл. Якщо обирається G-code, система використовує параметри друку і формує керуючі інструкції для FDM-принтера. Інтерфейс повинен чітко показувати користувачу, який саме файл буде отримано і для чого його можна використати.

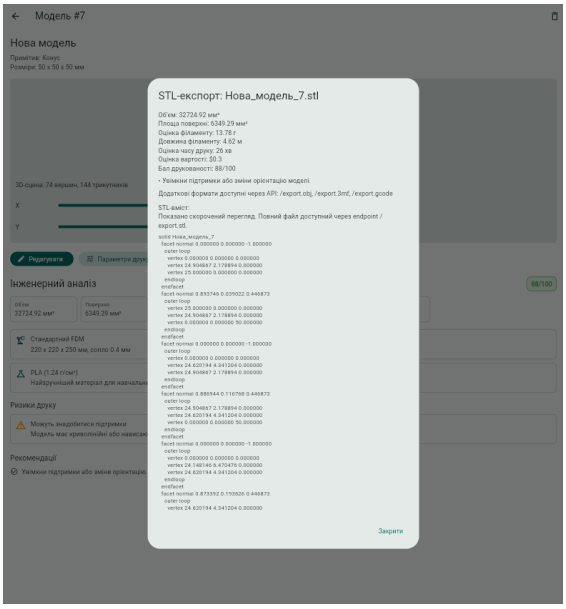


Рис. 3.6 Екран експорту моделі та генерації G-code

Загальну логіку переходів між екранами доцільно подати у вигляді графа діалогів. Такий граф показує, як користувач переходить від авторизації до

головного екрана, списку моделей, створення моделі, деталей, параметрів друку, аналізу, перегляду та експорту. Граф діалогів є корисним для проєктування, оскільки дозволяє побачити не окремі екрани, а всю структуру взаємодії користувача із застосунком.

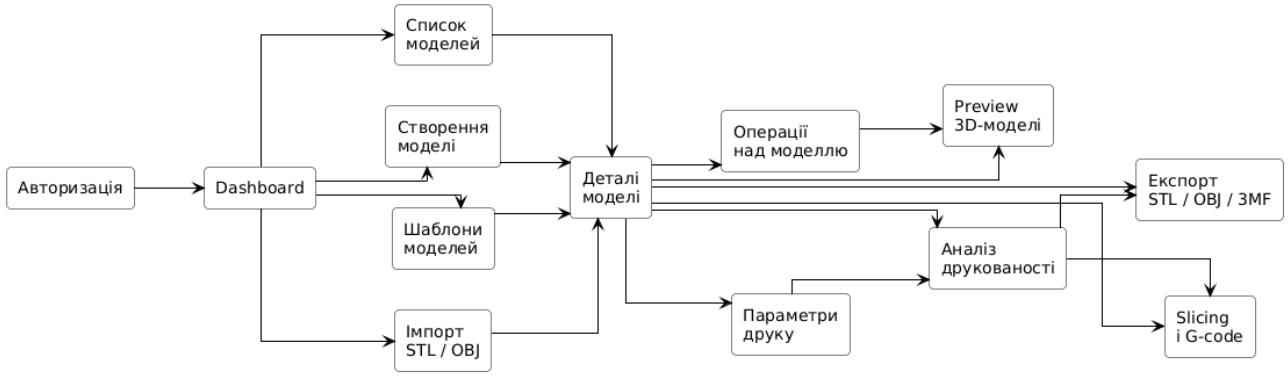


Рис. 3.7 Граф діалогів мобільного застосунку

Під час проєктування інтерфейсу також потрібно враховувати обробку помилок. Якщо користувач вводить некоректні параметри моделі, не заповнює обов’язкові поля, обирає файл непідтримуваного формату або запускає аналіз без необхідних даних, система повинна показати зрозуміле повідомлення. Повідомлення про помилки мають бути короткими й конкретними, щоб користувач розумів, яку дію потрібно виправити. Це особливо важливо для початківців, які можуть не знати технічних причин проблеми.

До основних принципів проєктування інтерфейсу розроблюваного застосунку можна віднести послідовність, простоту, мінімальну кількість зайвих дій, зрозуміле групування параметрів і наявність візуального контролю результату. Кожен екран має відповідати конкретному етапу роботи з моделлю. Авторизація відкриває доступ до системи, список моделей дозволяє обрати об’єкт, екран створення відповідає за формування моделі, екран параметрів — за підготовку до друку, екран аналізу — за оцінку результату, а екран експорту — за отримання файлу.

Узагальнено інтерфейс мобільного застосунку будується навколо робочого сценарію “модель → параметри → аналіз → експорт”. Така структура відповідає

логіці FDM-друку та дозволяє користувачу не втрачати послідовність дій. Це відрізняє застосунок від професійних CAD-систем, де кількість інструментів може бути значно більшою, але для початківця складнішою для освоєння.

Таким чином, проєктування інтерфейсу користувача спрямоване на створення зрозумілого мобільного середовища для базового моделювання та підготовки 3D-моделей до FDM-друку. Інтерфейс повинен забезпечити доступ до основних функцій системи, підтримувати логічну навігацію між екранами, надавати користувачу візуальний контроль моделі та допомагати послідовно виконати всі етапи підготовки до друку. Саме така структура інтерфейсу відповідає цільовій аудиторії та загальній меті бакалаврської роботи.

Після завершення етапу проєктування наступним кроком є реалізація програмного продукту. Якщо у попередньому розділі було визначено архітектуру системи, вимоги, сценарії використання, структуру бази даних та логіку інтерфейсу, то в цьому розділі розглядається безпосереднє створення мобільного застосунку та серверної частини. Основна увага приділяється тому, як спроєктовані функції були реалізовані у програмному коді та як окремі модулі взаємодіють між собою.

Розроблюваний програмний продукт реалізовано як клієнт-серверну систему. Мобільний клієнт створено з використанням Flutter та мови програмування Dart. Він відповідає за інтерфейс користувача, навігацію між екранами, введення параметрів моделей, відображення результатів аналізу, попередній перегляд і взаємодію з серверною частиною. Серверну частину реалізовано мовою Python із використанням FastAPI. Backend відповідає за обробку запитів, роботу з базою даних, виконання розрахунків, створення моделей, аналіз друкованості, імпорт, експорт і генерацію G-code.

У межах реалізації було створено функціональність, яка відповідає основному сценарію роботи користувача: авторизація, перегляд списку моделей, створення параметричної моделі, використання шаблонів, імпорт STL/OBJ-файлів, налаштування параметрів друку, виконання аналізу друкованості, попередній перегляд моделі, експорт у формати STL, OBJ, 3MF та формування G-code для

FDM-принтера. Така послідовність дозволяє користувачу виконати базову підготовку моделі до друку в одному програмному середовищі.

Важливою особливістю реалізації є те, що застосунок не подається як промислова CAD-система або професійний slicer. Його функціональність орієнтована на навчально-прикладний рівень, тому основний акцент зроблено на зрозумілому інтерфейсі, базовому параметричному моделюванні, роботі з типовими примітивами, аналізі основних показників друку та підтримці поширених форматів. Такий підхід відповідає цільовій аудиторії — студентам, початківцям і користувачам навчальних FDM-принтерів.

Серверна частина реалізує основну логіку обробки даних. У ній передбачено маршрути для авторизації, роботи з матеріалами, профілями принтерів, шаблонами, моделями, операціями над геометрією, аналізом, імпортом, експортом, slicing та генерацією G-code. Для збереження даних використовується SQLite, що є достатнім рішенням для навчально-прикладного MVP. База даних зберігає інформацію про користувачів, моделі, параметри друку, матеріали, профілі принтерів та інші дані, необхідні для функціонування застосунку.

Мобільний клієнт забезпечує користувацьку взаємодію з системою. У ньому реалізовано екрани авторизації, dashboard, списку моделей, створення моделі, деталей моделі, налаштування параметрів друку, аналізу, перегляду та експорту. Користувач працює з моделлю через зрозумілу послідовність дій, а складніші операції обробки виконуються на серверній частині. Це дозволяє не перевантажувати мобільний застосунок зайвою логікою та зберігати чіткий розподіл відповідальності між клієнтом і backend.

У цьому розділі буде розглянуто вибір програмних засобів реалізації, структуру серверної частини, реалізацію мобільного клієнта, модулі параметричного 3D-моделювання, аналізу друкованості, імпорту, експорту та генерації G-code. Також буде описано роботу основних екранів застосунку та зв’язок реалізованих функцій із поставленими вимогами. Це дозволить показати, яким чином спроєктована система була втілена у вигляді працездатного програмного продукту.

3.2 Вибір програмних засобів реалізації

Вибір програмних засобів реалізації є важливим етапом створення мобільного застосунку, оскільки від нього залежить структура проєкту, спосіб взаємодії між модулями, зручність подальшого супроводу та можливість реалізації функцій, визначених у попередніх розділах. Для розроблюваної системи було обрано стек технологій, який дозволяє реалізувати мобільний клієнт, серверну частину, базу даних, REST API, обробку параметрів 3D-моделей, аналіз друкованості, імпорт, експорт і генерацію G-code.

Під час вибору технологій враховувалося, що програмний продукт має бути навчально-прикладним MVP, а не промисловою CAD-системою. Тому основними критеріями були простота розробки, достатня продуктивність для базових операцій, зручність створення мобільного інтерфейсу, можливість організувати клієнт-серверну взаємодію та підтримати роботу з файлами моделей. Обраний стек має відповідати реальним функціям застосунку: авторизації, роботі з моделями, параметрами друку, матеріалами, принтерами, аналізом і експортом.

Для реалізації мобільного клієнта використано Flutter та мову програмування Dart. Flutter дозволяє створювати мобільні інтерфейси з єдиною кодовою базою та зручно будувати екрани, форми, списки, кнопки, картки моделей і навігацію між сторінками. Для цього проєкту Flutter є доцільним вибором, оскільки застосунок має велику кількість інтерфейсних елементів: екран авторизації, dashboard, список моделей, сторінку створення моделі, сторінку деталей, блоки параметрів друку, аналіз друкованості, preview та експорт.

Мова Dart використовується для опису логіки мобільного клієнта, обробки дій користувача, формування запитів до серверної частини та відображення отриманих результатів. У межах застосунку Dart-код відповідає за організацію екранів, перевірку введених параметрів, взаємодію з REST API, обробку відповідей сервера та оновлення стану інтерфейсу. Завдяки цьому мобільна частина працює як зручний інструмент керування процесом створення та підготовки моделі.

Для реалізації серверної частини використано Python та фреймворк FastAPI. Python є зручним для швидкої розробки backend-логіки, роботи з файлами, обчислень, обробки структурованих даних і реалізації алгоритмічних модулів. У цьому проєкті серверна частина виконує ключові операції: приймає запити від мобільного клієнта, працює з базою даних, створює моделі, обробляє параметри друку, виконує аналіз друкованості, імпортує STL/OBJ-файли, формує STL, OBJ, 3MF та G-code.

FastAPI обрано як основу backend-частини, оскільки він добре підходить для побудови REST API. За допомогою FastAPI у проєкті реалізовано маршрути авторизації, роботи з користувачами, матеріалами, принтерами, шаблонами, моделями, геометрією, аналізом, імпортом, експортом і slicing. Такий підхід дозволяє чітко розділити функціональність за групами маршрутів і забезпечити зрозумілу взаємодію між мобільним клієнтом та сервером.

Для збереження даних використано SQLite. У межах бакалаврської роботи це рішення є достатнім, оскільки система розглядається як навчально-прикладний MVP, а не як високонавантажений промисловий сервіс. SQLite дозволяє зберігати користувачів, моделі, параметри друку, матеріали, профілі принтерів, шаблони та службові дані без необхідності розгортання окремого серверу баз даних. Це спрощує запуск, тестування та демонстрацію програмного продукту.

Для обміну даними між мобільним клієнтом і серверною частиною використовується REST API. Такий підхід дає змогу відокремити інтерфейс від серверної логіки. Мобільний застосунок надсилає HTTP-запити, а backend повертає структуровані відповіді, які потім відображаються користувачу. Наприклад, під час створення моделі клієнт передає тип примітива та параметри, сервер створює модель і повертає результат. Під час аналізу клієнт надсилає запит для конкретної моделі, а сервер повертає розраховані показники.

Для валідації даних у серверній частині використовуються моделі даних, які дозволяють перевіряти структуру вхідних запитів і формувати передбачувані відповіді. Це важливо для застосунку, де користувач вводить числові параметри

моделі, параметри друку, матеріали та інші дані. Перевірка таких даних зменшує ймовірність помилок під час створення моделі, аналізу або експорту.

Окреме значення мають засоби роботи з файлами 3D-моделей. У проєкті передбачено імпорт ASCII STL та OBJ, а також експорт STL, OBJ, 3MF і G-code. Це потребує реалізації модулів, які можуть формувати або обробляти геометричні дані, зберігати інформацію про модель і перетворювати її у відповідний формат. Для навчально-прикладного застосунку така функціональність є важливою, оскільки вона демонструє повний базовий шлях від цифрової моделі до файлу, придатного для подальшої підготовки або друку.

Обраний стек технологій відповідає архітектурі програмного продукту. Flutter і Dart забезпечують реалізацію мобільного клієнта, FastAPI і Python — серверну логіку, SQLite — збереження даних, а REST API — зв’язок між клієнтом і backend. Такий набір засобів дозволяє реалізувати всі основні функції застосунку без надмірного ускладнення технічної структури.

Важливо підкреслити, що використання конкретних технологій саме по собі не є практичною новизною роботи. Їх роль полягає у забезпеченні реалізації функціональності, яка була обґрунтована в попередніх розділах. Практична цінність програмного продукту полягає не у виборі Flutter, FastAPI або SQLite окремо, а в тому, що за допомогою цих засобів створено мобільний застосунок, який поєднує параметричне моделювання, аналіз друкованості, роботу з форматами STL/OBJ/3MF і генерацію G-code для FDM-друку.

Таким чином, обрані програмні засоби реалізації є доцільними для поставленої задачі. Вони дозволяють побудувати клієнт-серверну систему, реалізувати зручний мобільний інтерфейс, організувати збереження даних, виконувати обробку моделей і підтримувати основні етапи підготовки 3D-моделі до FDM-друку.

3.3 Реалізація серверної частини застосунку

Серверна частина розроблюваного застосунку реалізована мовою Python із використанням фреймворку FastAPI. Вона є центральним рівнем системи, який приймає запити від мобільного клієнта, обробляє дані, взаємодіє з базою даних SQLite, виконує розрахунки, формує результати аналізу друкованості та забезпечує експорт моделей у потрібні формати. Саме backend-частина відповідає за більшість логіки, пов’язаної з 3D-моделями та підготовкою до FDM-друку.

У структурі проєкту серверна частина зосереджена у файлі main.py, де описано основні моделі даних, маршрути API, логіку роботи з користувачами, моделями, матеріалами, профілями принтерів, шаблонами, аналізом, імпортом, експортом та генерацією G-code. Такий підхід відповідає рівню навчально-прикладного MVP: система має достатню функціональність для демонстрації повного базового сценарію, але не ускладнюється надмірною кількістю окремих сервісів.

Серверна частина побудована навколо REST API. Мобільний клієнт надсилає HTTP-запити до відповідних маршрутів, а сервер повертає структуровані відповіді. Це дозволяє відокремити інтерфейс користувача від внутрішньої логіки обробки моделей. Наприклад, коли користувач створює модель у мобільному застосунку, клієнт передає на сервер тип примітива та його параметри. Сервер перевіряє отримані дані, створює модель, зберігає її у базі даних і повертає клієнту результат.

У серверній частині реалізовано модуль авторизації користувачів. Для входу до системи використовується маршрут /auth/login, а для отримання інформації про поточного користувача — /auth/me. У системі передбачено ролі користувачів, зокрема admin і student. Такий поділ дозволяє розмежовувати звичайне використання застосунку та адміністративні або демонстраційні можливості. Авторизація є важливою частиною backend-логіки, оскільки моделі й параметри повинні бути пов’язані з конкретним користувачем.

Для роботи з 3D-моделями реалізовано набір маршрутів, які забезпечують створення, перегляд, редагування та видалення моделей. Основні операції

виконуються через групу маршрутів /models. Модель у системі може бути створена як параметричний об’єкт або імпортована з файлу. Для параметричних моделей зберігаються назва, тип геометричного примітива, розміри та інші характеристики. Така структура дозволяє надалі виконувати аналіз, перегляд, експорт і генерацію файлів для друку.

Окремо реалізовано роботу з шаблонами моделей. Сервер надає список доступних шаблонів через маршрут /templates, а створення моделі на основі конкретного шаблону виконується через /templates/{id}/create. Ця функціональність важлива для навчального сценарію, оскільки користувач може швидко створити типову модель без ручного введення всіх параметрів. Шаблони скорочують кількість дій користувача та роблять застосунок зручнішим для демонстраційних і практичних задач.

Для налаштування друку серверна частина містить маршрути роботи з матеріалами та профілями принтерів. Маршрут /materials повертає доступні матеріали, наприклад PLA, ABS і PETG, а маршрут /printers — профілі FDM-принтерів. Ці дані використовуються під час аналізу друкованості та генерації G-code. Матеріал впливає на розрахунок маси, вартості й температурних параметрів, а профіль принтера потрібний для врахування робочої області та технічних обмежень обладнання.

Важливою частиною backend-реалізації є модуль аналізу друкованості. Він доступний через маршрут /analysis і використовується для попередньої оцінки моделі перед друком. Серверна частина розраховує основні показники: об’єм моделі, площу поверхні, орієнтовну масу матеріалу, довжину філаменту, приблизний час друку та вартість. Такі розрахунки допомагають користувачу зрозуміти, як параметри моделі та друку впливають на результат.

Для роботи з геометрією моделей у backend-частині передбачено маршрути /geometry та /operations. Вони пов’язані з базовими операціями над моделлю, такими як масштабування, обертання, переміщення, дзеркальне відображення, створення отвору та деформаційні перетворення. У межах навчально-прикладного

застосунку ці операції дають змогу показати базову логіку редагування моделі без використання професійного CAD-середовища.

Сервер також підтримує імпорт готових моделей через маршрут /models/import. У межах проєкту передбачено імпорт ASCII STL та OBJ-файлів. Після завантаження файл обробляється серверною частиною, модель зберігається в системі та стає доступною для подальшого перегляду, аналізу або експорту. Це розширює можливості застосунку, оскільки користувач може працювати не лише з моделями, створеними в самому застосунку, а й із зовнішніми файлами.

Експорт реалізовано через групу маршрутів /export, /export.stl, /export.obj, /export.3mf та /export.gcode. Формати STL, OBJ і 3MF використовуються для збереження або передавання моделі, а G-code — для формування керуючих інструкцій для FDM-принтера. Окремо передбачено маршрут /slice, який відповідає за поділ моделі на шари. Це дозволяє реалізувати повний базовий сценарій: модель → параметри друку → slicing → G-code.

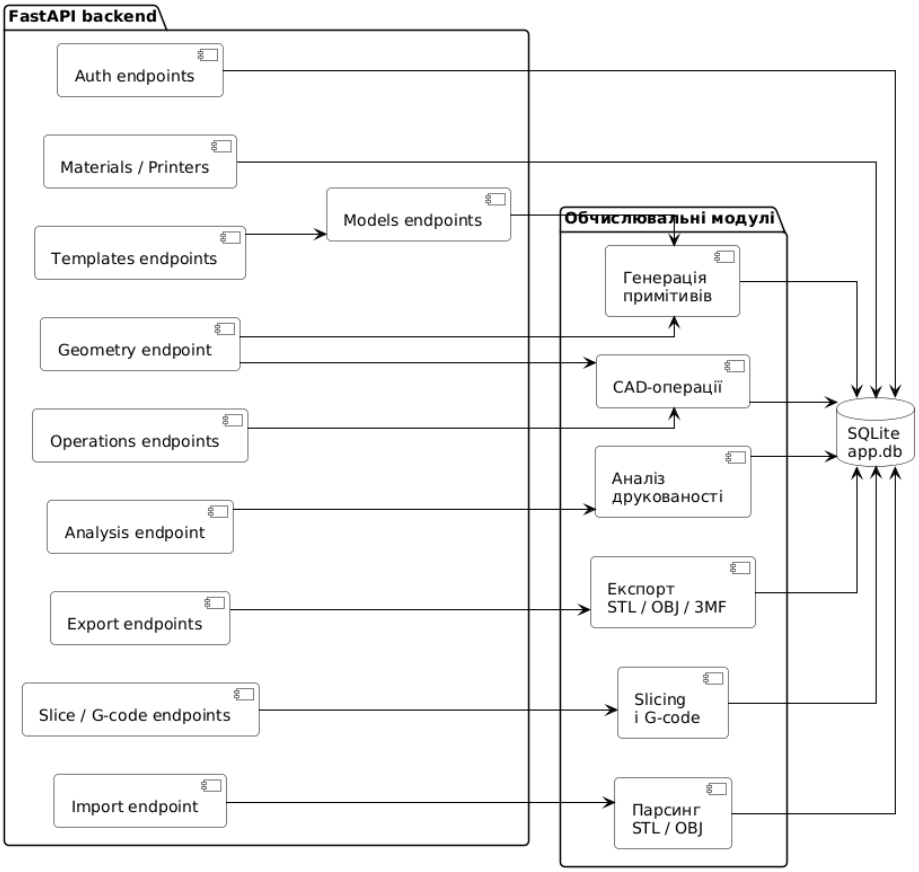


Рис. 3.8 Структура серверної частини мобільного застосунку

На рисунку 3.8 подано узагальнену структуру серверної частини. FastAPI backend приймає запити від мобільного клієнта, спрямовує їх до відповідних модулів, взаємодіє з базою даних SQLite та повертає користувачу результати обробки, аналізу або сформовані файли.

Для збереження даних використовується SQLite. У базі даних зберігаються користувачі, моделі, параметри друку, матеріали, профілі принтерів, шаблони та службова інформація. SQLite є доцільним вибором для цього проєкту, оскільки застосунок розглядається як MVP і не потребує складної серверної інфраструктури баз даних. При цьому SQLite дозволяє зберігати всі необхідні дані для демонстрації повного циклу роботи системи.

Окремим елементом серверної частини є dashboard-модуль, доступний через маршрут /dashboard. Він надає узагальнену інформацію про стан системи, моделі, параметри або інші показники, які можуть бути відображені у мобільному клієнті. Такий модуль робить застосунок більш інформативним і дозволяє користувачу швидко оцінити поточний стан роботи з моделями.

Реалізація backend-частини побудована так, щоб основна логіка роботи з моделями виконувалася на сервері, а мобільний клієнт залишався зручним інтерфейсом для керування процесом. Це особливо важливо для операцій, пов’язаних з аналізом, експортом і генерацією G-code, оскільки вони потребують обробки даних і формування файлів. Завдяки такому поділу відповідальності система залишається логічно структурованою.

Таким чином, серверна частина застосунку реалізує ключову логіку програмного продукту. Вона забезпечує авторизацію, роботу з користувачами, моделями, матеріалами, профілями принтерів, шаблонами, геометрією, аналізом друкованості, імпортом, експортом, slicing та генерацією G-code. Така реалізація відповідає поставленим задачам бакалаврської роботи та забезпечує технічну основу для роботи мобільного клієнта.

3.4 Реалізація мобільного клієнта

Мобільний клієнт розроблюваного застосунку реалізовано з використанням Flutter та мови програмування Dart. Він виконує роль інтерфейсного рівня системи, через який користувач взаємодіє з основними функціями програмного продукту: авторизацією, переглядом моделей, створенням параметричних об’єктів, вибором шаблонів, налаштуванням параметрів друку, аналізом друкованості, попереднім переглядом та експортом результатів.

Основне завдання мобільного клієнта полягає не у виконанні складних обчислень, а в організації зручної взаємодії користувача із серверною частиною. Клієнт приймає введені дані, надсилає їх до FastAPI backend через REST API, отримує відповіді сервера та відображає результати у зрозумілому вигляді. Такий підхід відповідає клієнт-серверній архітектурі застосунку, де мобільна частина відповідає за інтерфейс і навігацію, а серверна — за обробку моделей, розрахунки, аналіз та формування файлів.

Структура мобільного клієнта побудована навколо основних екранів застосунку. Після запуску користувач потрапляє на екран авторизації, де вводить облікові дані. Після успішного входу відкривається основна частина застосунку, яка містить dashboard, список моделей та доступ до функцій створення, імпорту, аналізу й експорту. Така структура дозволяє користувачу рухатися логічним шляхом: спочатку отримати доступ до системи, потім обрати або створити модель, налаштувати її параметри та виконати необхідні дії для підготовки до друку.

Екран авторизації реалізовано як початкову точку роботи із застосунком. На ньому користувач вводить логін і пароль, після чого мобільний клієнт надсилає запит до серверного маршруту авторизації. У разі успішної відповіді користувач переходить до головного екрана. Якщо дані введено неправильно або сервер повертає помилку, застосунок має відобразити відповідне повідомлення. Такий механізм потрібний для контролю доступу до моделей і даних користувача.

Після входу користувач переходить до головного екрана, де відображається загальна інформація про роботу із застосунком. Dashboard-модуль

використовується для короткого представлення стану системи: кількості моделей, доступних шаблонів, матеріалів або інших узагальнених показників. Такий екран не є обов’язковим для самого процесу друку, але робить застосунок зручнішим і дозволяє користувачу швидко зорієнтуватися у власних даних.

Одним із центральних екранів мобільного клієнта є список моделей. На ньому користувач бачить створені або імпортовані 3D-моделі та може перейти до детальної інформації про кожну з них. Для кожної моделі доцільно відображати назву, тип, короткі параметри та доступні дії. Саме з цього екрана користувач починає основну роботу: відкриває існуючу модель, створює нову, імпортує файл або переходить до аналізу.

Екран створення моделі реалізує функцію базового параметричного моделювання. Користувач обирає тип геометричного примітива — куб, циліндр, сферу, конус, піраміду або тор — і задає числові параметри моделі. Після підтвердження дані передаються на сервер, де створюється відповідний запис моделі. Такий підхід дозволяє користувачу створити просту 3D-модель без використання складного професійного CAD-інтерфейсу.

Окремо в мобільному клієнті передбачено роботу з шаблонами. Користувач може обрати готову модельну заготовку та створити на її основі новий об’єкт. Це зручно для навчальних задач, коли потрібно швидко отримати типову модель без повторного введення всіх параметрів. Мобільний клієнт надсилає серверу запит на створення моделі з вибраного шаблону, після чого новий об’єкт з’являється у списку моделей.

Екран деталей моделі призначений для перегляду основної інформації про вибраний об’єкт. На ньому відображаються назва моделі, тип, розміри, параметри друку, матеріал, профіль принтера та доступні операції. З цього екрана користувач може перейти до редагування параметрів, аналізу друкованості, попереднього перегляду або експорту. Таким чином, екран деталей виконує роль центру керування конкретною моделлю.

Для налаштування FDM-друку в мобільному клієнті реалізовано введення технологічних параметрів. Користувач може обрати матеріал, висоту шару,

відсоток заповнення, швидкість друку, температуру сопла, температуру друкованого столу, діаметр сопла та потребу в підтримках. Ці дані передаються до серверної частини і використовуються під час аналізу друкованості та генерації G-code. У мобільному інтерфейсі такі параметри мають бути згруповані зрозуміло, щоб користувач бачив їх не як складний набір технічних полів, а як частину підготовки моделі до друку.

Мобільний клієнт також відображає результати аналізу друкованості. Після запуску аналізу застосунок надсилає запит на сервер і отримує розраховані показники: об’єм, площу поверхні, орієнтовну масу матеріалу, довжину філаменту, час друку та вартість. Ці дані подаються користувачу у вигляді зрозумілих числових показників. Завдяки цьому користувач може оцінити модель ще до експорту або запуску друку.

Окреме місце займає попередній перегляд моделі. У застосунку передбачено візуальне представлення об’єкта, яке дає змогу оцінити форму моделі та правильність заданих параметрів. Для навчально-прикладного MVP важливо, щоб користувач міг не лише вводити числові значення, а й бачити результат їх застосування. Саме тому preview-модуль є важливою частиною мобільного клієнта.

Завершальним етапом роботи користувача в мобільному клієнті є експорт моделі або генерація G-code. На відповідному екрані користувач обирає потрібний формат: STL, OBJ, 3MF або G-code. Після цього мобільний клієнт надсилає запит до серверної частини, а сервер формує потрібний файл. Користувач отримує результат, який може бути використаний для подальшої роботи з моделлю або підготовки до друку.

Важливою частиною реалізації мобільного клієнта є обробка помилок. Якщо сервер повертає повідомлення про некоректні параметри, помилку авторизації, проблему імпорту або неможливість виконати аналіз, застосунок повинен показати користувачу зрозуміле повідомлення. Це особливо важливо для цільової аудиторії, оскільки студенти й початківці можуть не мати достатнього досвіду для самостійного визначення причини помилки.

Таким чином, мобільний клієнт забезпечує основну взаємодію користувача із системою. Він об’єднує екрани авторизації, dashboard, списку моделей, створення моделі, шаблонів, деталей моделі, параметрів друку, аналізу, preview та експорту. Його реалізація дозволяє користувачу пройти повний базовий сценарій підготовки 3D-моделі до FDM-друку в одному мобільному середовищі, а складні операції обробки даних залишаються на рівні серверної частини.

3.5 Реалізація модуля параметричного 3D-моделювання

Модуль параметричного 3D-моделювання є одним із ключових функціональних блоків розроблюваного застосунку, оскільки саме він забезпечує створення базових тривимірних об’єктів без використання складного професійного CAD-середовища. Його призначення полягає в тому, щоб користувач міг швидко сформувати просту модель, задавши її тип і основні числові параметри.

У межах застосунку параметричне моделювання реалізовано для базових геометричних примітивів: куба, циліндра, сфери, конуса, піраміди та тора. Такий набір об’єктів є достатнім для навчально-прикладного MVP, оскільки дозволяє продемонструвати принцип створення 3D-моделі через параметри, а також показати зв’язок між геометрією моделі, її розмірами, об’ємом і подальшою підготовкою до FDM-друку.

Основна логіка роботи модуля полягає в тому, що користувач у мобільному клієнті обирає тип моделі та вводить потрібні значення: ширину, висоту, глибину, радіус, кількість сегментів або інші параметри залежно від вибраного примітива. Після підтвердження введені дані передаються на серверну частину через REST API. Backend перевіряє коректність параметрів, створює модель відповідного типу та зберігає її у базі даних.

Для параметричних моделей важливо, що вони описуються не лише як готова геометрія, а як набір вихідних характеристик. Наприклад, куб може описуватися через ширину, висоту і глибину, циліндр — через радіус і висоту, сфера — через радіус, а тор — через основний і внутрішній радіуси. Такий підхід дає змогу легко

змінювати параметри моделі та повторно виконувати розрахунки без повного ручного перемоделювання об’єкта.

У серверній частині модель зберігається разом із назвою, типом примітива, параметрами, датою створення та зв’язком із користувачем. Це дозволяє надалі відкривати модель у мобільному клієнті, змінювати її параметри, виконувати аналіз друкованості, формувати попередній перегляд і експортувати результат у потрібному форматі. Таким чином, параметрична модель стає центральним об’єктом, з яким працюють інші модулі системи.

Окрім створення моделей з нуля, у застосунку передбачено використання шаблонів. Шаблон є попередньо підготовленою конфігурацією моделі, яку користувач може використати як основу для нового об’єкта. Такий підхід є зручним для навчальних задач, коли потрібно швидко створити типову модель, наприклад калібрувальний куб, простий корпус або технічну форму. Після вибору шаблону сервер створює нову модель на його основі, а користувач може далі змінювати її параметри.

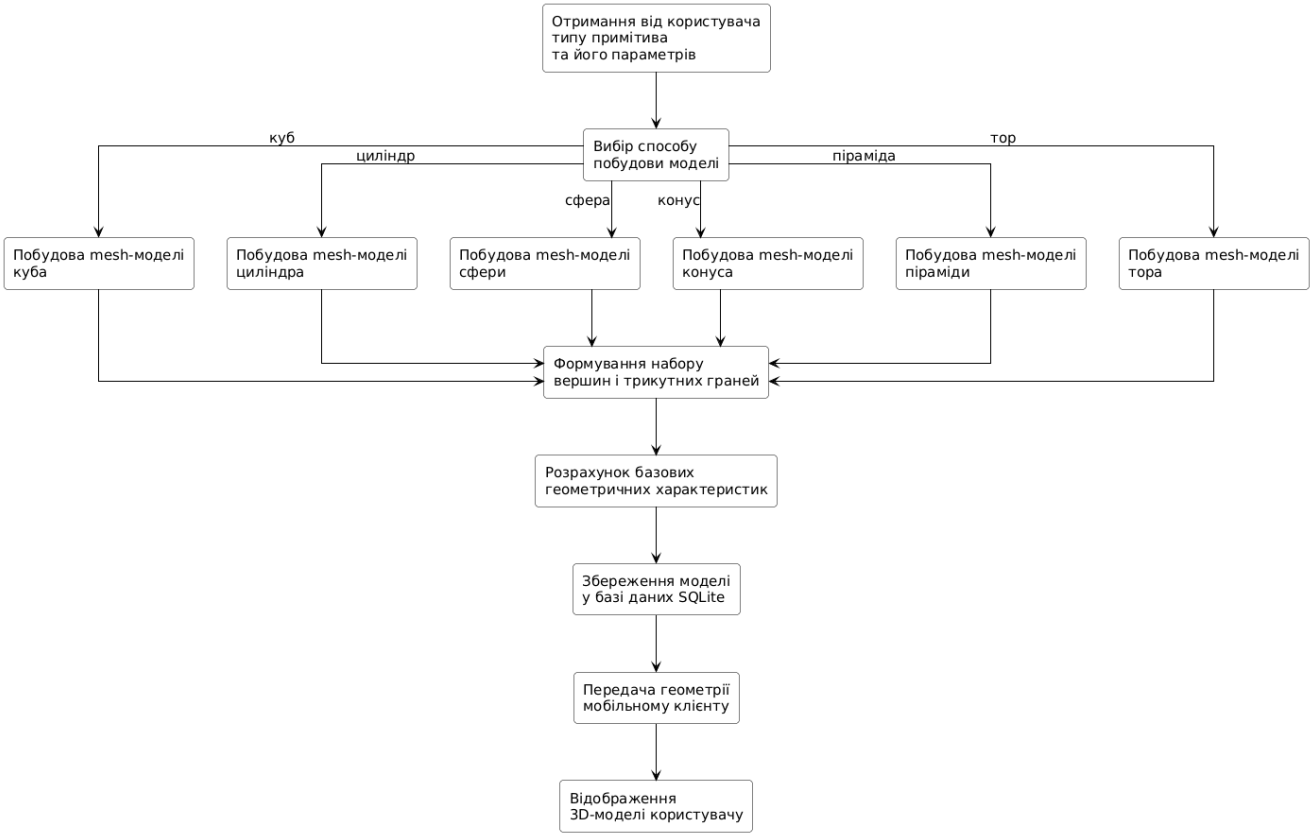


Рис. 3.9 Схема створення параметричної 3D-моделі у мобільному застосунку

На рисунку 3.9 показано послідовність створення параметричної моделі: вибір примітива, введення параметрів, передача даних на сервер, створення моделі, збереження в базі даних і відображення результату в мобільному клієнті. Така схема демонструє, що користувач працює з простими параметрами, а основна логіка формування моделі виконується на рівні серверної частини.

Реалізація параметричного моделювання має важливе значення для всієї системи, оскільки саме вона забезпечує початковий етап роботи користувача з 3D-об’єктом. Без цього модуля застосунок виконував би лише функції перегляду або обробки готових файлів. Наявність параметричного створення моделей дозволяє користувачу самостійно формувати базову геометрію та одразу переходити до її аналізу, налаштування параметрів друку й експорту.

Таким чином, модуль параметричного 3D-моделювання реалізує основну ідею застосунку — спростити створення простих моделей для FDM-друку. Він не замінює професійні CAD-системи, але надає користувачу достатній набір інструментів для базового навчального сценарію: створити модель, змінити її параметри, переглянути результат і підготувати об’єкт до подальшого друку.

3.6 Реалізація модуля аналізу друкованості

Модуль аналізу друкованості призначений для попередньої оцінки 3D-моделі перед її експортом або генерацією G-code. Його основна задача полягає в тому, щоб користувач міг ще до запуску друку отримати базову інформацію про модель: орієнтовний об’єм, площу поверхні, масу матеріалу, довжину філаменту, час друку та приблизну вартість виготовлення. Це допомагає краще оцінити доцільність друку та за потреби змінити параметри моделі або налаштування FDM-друку.

У розроблюваному застосунку аналіз друкованості виконується на серверній частині. Мобільний клієнт надсилає запит для конкретної моделі, а backend отримує потрібні дані з бази: геометричні параметри, матеріал, профіль принтера та налаштування друку. Після цього сервер виконує розрахунки й повертає результат у мобільний застосунок для відображення користувачу.

Для параметричних моделей аналіз базується на їхніх числових характеристиках. Наприклад, для куба або прямокутного об’єкта враховуються ширина, висота й глибина, для циліндра — радіус і висота, для сфери — радіус. На основі цих значень можна визначити приблизний об’єм і площу поверхні. Для імпортованих STL/OBJ-моделей аналіз може виконуватися на основі mesh-даних, тобто геометричного представлення моделі у вигляді вершин і граней.

Важливим елементом аналізу є врахування матеріалу. У системі передбачено використання матеріалів PLA, ABS і PETG. Кожен із них має власну густину, температурні параметри та орієнтовну вартість. Саме густина матеріалу використовується для розрахунку маси моделі, а ціна матеріалу — для приблизної оцінки вартості друку. Такий підхід дозволяє користувачу побачити, як вибір матеріалу впливає на результат ще до фактичного друку.

Параметри друку також впливають на результати аналізу. Висота шару, відсоток заповнення, швидкість друку, діаметр сопла, температура та наявність підтримок визначають орієнтовний час виготовлення, витрату матеріалу та складність друку. Наприклад, більший відсоток заповнення збільшує масу та вартість моделі, а менша висота шару може збільшити тривалість друку. Тому аналіз друкованості повинен враховувати не лише геометрію, а й технологічні налаштування.

У межах реалізації модуль аналізу виконує кілька послідовних дій: отримує модель, перевіряє наявність необхідних параметрів, визначає об’єм і площу поверхні, враховує матеріал та параметри друку, обчислює додаткові показники і формує результат для клієнта. Якщо даних недостатньо або модель має некоректні параметри, система повинна повернути повідомлення про помилку.

3.7 Реалізація імпорту, експорту та генерації G-code

Модуль імпорту, експорту та генерації G-code забезпечує завершальний етап роботи користувача з 3D-моделлю. Якщо модуль параметричного моделювання відповідає за створення об’єкта, а модуль аналізу друкованості — за попередню

оцінку його параметрів, то цей модуль дозволяє отримати результат у вигляді файлу, який можна використати в інших програмах або в процесі підготовки до FDM-друку.

У застосунку реалізовано імпорт моделей у форматах STL та OBJ. Ця можливість потрібна для того, щоб користувач міг працювати не лише з моделями, створеними безпосередньо в мобільному застосунку, а й з готовими файлами, підготовленими в інших середовищах. Під час імпорту користувач обирає файл, мобільний клієнт передає його на серверну частину, а backend перевіряє формат, виконує базову обробку та зберігає інформацію про модель у системі.

Формат STL використовується для представлення геометрії моделі у вигляді трикутної сітки. У межах реалізації застосунок підтримує імпорт ASCII STL-файлів. Такий формат є зручним для навчально-прикладного MVP, оскільки його структура є зрозумілою для обробки та дозволяє отримати базові геометричні дані моделі. Формат OBJ також використовується для опису полігональної моделі, зокрема вершин і граней. Його підтримка розширює можливості застосунку та дозволяє працювати з файлами, створеними в різних 3D-редакторах.

Після імпорту модель стає доступною в загальному списку моделей. Користувач може відкрити її, переглянути, виконати аналіз друкованості або експортувати в інший формат. Такий підхід робить застосунок гнучкішим, оскільки він не обмежується лише параметричними примітивами та шаблонами.

Експорт моделей реалізовано у форматах STL, OBJ та 3MF. Формати STL і OBJ використовуються для збереження геометрії моделі та подальшої роботи з нею в інших програмах. Формат 3MF є сучаснішим форматом для задач 3D-друку, оскільки може містити більше інформації про модель і краще відповідає потребам обміну даними між системами підготовки до друку. Підтримка кількох форматів експорту дозволяє користувачу обрати той варіант, який найбільше відповідає подальшому сценарію використання.

Окремим результатом роботи застосунку є генерація G-code. На відміну від STL, OBJ або 3MF, G-code не описує модель як геометричний об’єкт, а містить керуючі інструкції для FDM-принтера. У ньому задаються переміщення

друкувальної головки, параметри подачі матеріалу, швидкість, температура та інші команди, необхідні для виконання друку. Саме тому генерація G-code є важливим етапом перетворення цифрової моделі у файл, придатний для практичного використання.

Перед формуванням G-code модель проходить етап slicing, тобто поділу на шари. У межах застосунку slicing реалізовано на базовому рівні, достатньому для навчально-прикладного сценарію. Серверна частина використовує дані про геометрію моделі, висоту шару, параметри друку, матеріал і профіль принтера, після чого формує послідовність команд. Така реалізація не є промисловим slicer-рішенням, але демонструє основний принцип перетворення 3D-моделі у набір керуючих інструкцій.

Рис. 3.10 Схема імпорту, експорту та генерації G-code у мобільному застосунку

Робота модуля імпорту та експорту тісно пов’язана з іншими частинами системи. Після імпорту модель зберігається в базі даних і може бути використана модулем аналізу друкованості. Під час експорту серверна частина використовує параметри моделі, а під час генерації G-code додатково враховуються матеріал, профіль принтера та технологічні налаштування друку. Завдяки цьому модуль не працює ізольовано, а є частиною загального сценарію підготовки моделі до FDM-друку.

Важливою частиною реалізації є перевірка вхідних і вихідних даних. Під час імпорту система повинна відхиляти файли непідтримуваних форматів або повідомляти користувача про помилку обробки. Під час експорту та генерації G-code необхідно перевіряти, чи існує модель, чи має вона достатньо параметрів і чи можна сформувати файл у вибраному форматі. Така обробка помилок підвищує стабільність застосунку та робить його зрозумілішим для користувача.

Таким чином, реалізація імпорту, експорту та генерації G-code забезпечує завершення основного робочого процесу в застосунку. Користувач може створити або завантажити модель, виконати її аналіз, а потім отримати результат у форматі STL, OBJ, 3MF або G-code. Це відповідає загальній меті програмного продукту — спростити базову підготовку 3D-моделей до FDM-друку в одному мобільному середовищі.

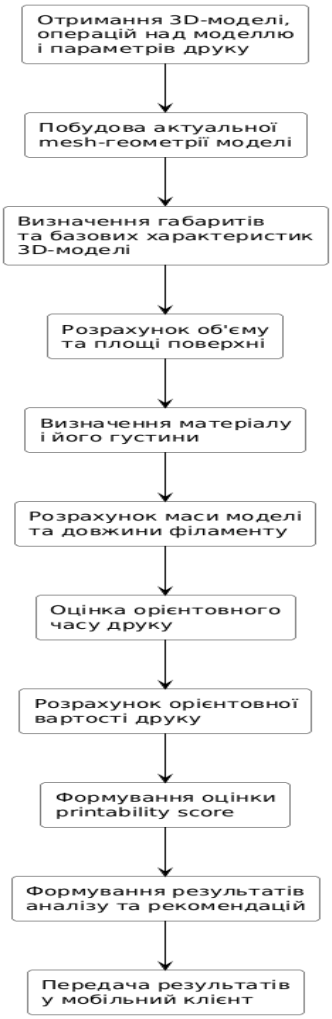


Рис. 3.11 Схема роботи модуля аналізу друкованості 3D-моделі

Результати аналізу відображаються в мобільному застосунку у вигляді зрозумілих показників. Користувач бачить не лише саму модель, а й практичну інформацію про її майбутній друк. Це особливо важливо для студентів і початківців, оскільки дає змогу простежити зв’язок між формою моделі, матеріалом, параметрами друку та кінцевими витратами.

Модуль аналізу друкованості не є заміною професійного інженерного аналізу або повноцінної перевірки моделі у спеціалізованому slicer-середовищі. Його

призначення полягає у виконанні базової попередньої оцінки, достатньої для навчального та початкового рівня роботи з FDM-друком. Така оцінка допомагає користувачу приймати прості практичні рішення: змінити розміри, обрати інший матеріал, зменшити заповнення або підготувати модель до експорту.

Таким чином, реалізація модуля аналізу друкованості є важливою частиною програмного продукту. Він забезпечує зв’язок між параметричним моделюванням і реальною підготовкою до друку, дає користувачу кількісну оцінку моделі та підтримує основну мету застосунку — спрощення процесу підготовки 3D-моделей до FDM-друку.

3.8 Опис роботи розробленого застосунку

Розроблений мобільний застосунок забезпечує послідовне виконання основних дій, необхідних для базового моделювання та підготовки 3D-моделі до FDM-друку. Його робота побудована навколо практичного сценарію, у якому користувач проходить шлях від входу в систему до створення або імпорту моделі, налаштування параметрів друку, аналізу друкованості, попереднього перегляду та експорту результату.

Початок роботи із застосунком здійснюється через авторизацію користувача. Користувач вводить облікові дані, після чого мобільний клієнт надсилає запит до серверної частини. У разі успішної перевірки користувач отримує доступ до основних функцій системи. Такий підхід дозволяє пов’язати створені моделі, параметри друку та результати аналізу з конкретним користувачем.

Після входу користувач переходить до основної частини застосунку, де може переглядати список моделей, працювати з dashboard-модулем, створювати нові об’єкти, використовувати шаблони або імпортувати готові STL/OBJ-файли. Список моделей є центральною точкою роботи, оскільки саме з нього користувач переходить до деталей конкретної моделі, аналізу, налаштувань або експорту.

Створення моделі може виконуватися двома основними способами. Перший спосіб — створення параметричної моделі з базового геометричного примітива.

Користувач обирає тип об’єкта, наприклад куб, циліндр, сферу, конус, піраміду або тор, після чого задає необхідні числові параметри. Другий спосіб — створення моделі на основі шаблону. У цьому випадку користувач обирає готову заготовку, а система створює на її основі новий об’єкт, який можна далі аналізувати або готувати до друку.

Окрім створення моделей у самому застосунку, реалізовано імпорт готових файлів у форматах STL та OBJ. Це дозволяє працювати з моделями, створеними в інших середовищах. Після імпорту модель зберігається в системі та стає доступною для подальшого перегляду, аналізу друкованості, експорту або генерації G-code.

Після вибору моделі користувач переходить до її параметрів. На цьому етапі задаються технологічні характеристики FDM-друку: матеріал, профіль принтера, висота шару, відсоток заповнення, швидкість друку, температура сопла, температура друкованого столу, діаметр сопла та потреба в підтримках. Ці параметри важливі, оскільки вони впливають на якість друку, витрату матеріалу, час виготовлення та вартість.

Наступним етапом є аналіз друкованості. Мобільний клієнт надсилає запит до серверної частини, а backend виконує розрахунки на основі геометрії моделі, матеріалу, профілю принтера та параметрів друку. У результаті користувач отримує орієнтовні показники: об’єм, площу поверхні, масу матеріалу, довжину філаменту, час друку та вартість. Такі дані допомагають оцінити модель до фактичного друку та за потреби змінити її параметри.

Попередній перегляд моделі використовується для візуального контролю результату. Користувач може оцінити форму об’єкта, його пропорції та відповідність заданим параметрам. Для навчально-прикладного застосунку це важливо, оскільки користувач бачить не лише числові значення, а й результат їх застосування до 3D-моделі.

Завершальним етапом роботи є експорт або генерація G-code. Користувач обирає потрібний формат результату: STL, OBJ, 3MF або G-code. Якщо обрано формат STL, OBJ чи 3MF, система формує файл моделі для подальшої роботи в

інших програмах. Якщо обрано G-code, серверна частина використовує модель і параметри друку для формування керуючих інструкцій для FDM-принтера.

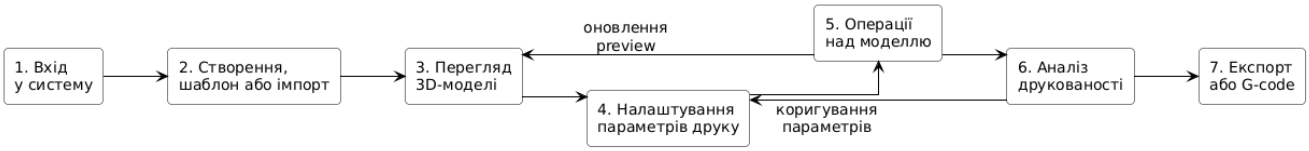


Рис 3.12 Узагальнений сценарій роботи розробленого застосунку

На рисунку 3.12 подано загальний порядок роботи користувача із застосунком. Послідовність охоплює авторизацію, вибір або створення моделі, налаштування параметрів FDM-друку, виконання аналізу, попередній перегляд і отримання результату у вигляді файлу моделі або G-code.

У процесі роботи застосунок також повинен коректно реагувати на помилкові ситуації. Наприклад, якщо користувач вводить некоректні розміри, не заповнює обов’язкові поля, обирає файл непідтримуваного формату або запускає аналіз для моделі з недостатніми параметрами, система має показати зрозуміле повідомлення. Це важливо для цільової аудиторії, оскільки застосунок орієнтований на студентів і початківців.

Таким чином, розроблений застосунок забезпечує повний базовий сценарій підготовки 3D-моделі до FDM-друку в одному мобільному середовищі. Користувач може створити або імпортувати модель, налаштувати параметри друку, виконати аналіз, переглянути результат і сформувати файл для подальшого використання. Така логіка роботи відповідає меті бакалаврської роботи — спрощенню процесу моделювання та підготовки 3D-моделей до друку.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Обґрунтування вибору видів тестування

Після реалізації мобільного застосунку необхідно виконати тестування, щоб перевірити його працездатність, відповідність поставленим вимогам і коректність виконання основних сценаріїв. Тестування є важливим етапом розробки, оскільки дозволяє виявити помилки в роботі інтерфейсу, серверної частини, бази даних, REST API, модулів аналізу, імпорту, експорту та генерації G-code.

Розроблений застосунок має клієнт-серверну архітектуру, тому тестування повинно охоплювати як мобільний клієнт, так і backend-частину. Мобільний клієнт перевіряється з погляду коректності роботи екранів, навігації, введення параметрів, відображення результатів і повідомлень про помилки. Серверна частина перевіряється через API-запити, роботу з базою даних, обробку моделей, виконання розрахунків, імпорт файлів, експорт моделей і формування G-code.

Основна увага під час тестування приділяється тим функціям, які безпосередньо пов’язані з метою бакалаврської роботи: створенню параметричних моделей, використанню шаблонів, імпорту STL/OBJ-файлів, налаштуванню параметрів FDM-друку, аналізу друкованості, попередньому перегляду, експорту STL/OBJ/3MF та генерації G-code. Саме ці функції забезпечують базовий сценарій підготовки 3D-моделі до друку.

У межах тестування потрібно перевірити не тільки сам факт наявності функцій, а й якість їх виконання. Наприклад, створення моделі повинно завершуватися збереженням коректних даних у базі, аналіз друкованості має повертати очікувані показники, імпорт повинен приймати лише підтримувані формати, а генерація G-code має формувати файл на основі параметрів моделі та друку. Такий підхід дозволяє оцінити систему не формально, а з погляду практичного використання.

Окремо необхідно перевірити нефункціональні вимоги, сформульовані на етапі проєктування. До них належать час виконання основних операцій, стабільність роботи застосунку, коректність обробки помилок, підтримка заявлених форматів, обмеження розміру файлів, захист доступу до функцій і збереження даних після виконання операцій. Наявність числових критеріїв дозволяє об’єктивно оцінити, чи відповідає програмний продукт очікуваним показникам.

Для тестування застосунку доцільно використати кілька видів перевірки: функціональне тестування, API-тестування, тестування інтерфейсу користувача, перевірку імпорту та експорту файлів, а також тестування нефункціональних вимог. Функціональне тестування дозволяє перевірити основні сценарії роботи користувача. API-тестування потрібне для перевірки backend-маршрутів. Тестування інтерфейсу дозволяє оцінити коректність переходів між екранами, введення даних і відображення результатів.

У цьому розділі буде обґрунтовано вибір видів тестування, розглянуто перевірку серверної частини, функціональне тестування мобільного застосунку, перевірку нефункціональних вимог, а також наведено тест-кейси та результати їх виконання. Це дозволить підтвердити, що розроблений застосунок виконує основні задачі, визначені у бакалаврській роботі, і може бути використаний як навчально-прикладний інструмент для базової підготовки 3D-моделей до FDM-друку.

4.2 Тестування серверної частини застосунку

Тестування серверної частини виконувалося для перевірки коректної роботи backend-рівня мобільного застосунку. Оскільки саме серверна частина відповідає за обробку запитів, роботу з базою даних, створення моделей, аналіз друкованості, імпорт, експорт і генерацію G-code, її перевірка є одним із ключових етапів тестування програмного продукту.

Серверну частину застосунку реалізовано з використанням Python та FastAPI. Вона приймає запити від мобільного клієнта через REST API, обробляє отримані дані, звертається до бази даних SQLite та повертає результат у структурованому вигляді. Тому під час тестування backend-частини перевірялися не лише окремі

маршрути API, а й логіка їх взаємодії з базою даних та функціональними модулями системи.

Основна увага приділялася перевірці маршрутів авторизації, роботи з моделями, матеріалами, профілями принтерів, шаблонами, аналізом друкованості, імпортом та експортом. Також перевірялося, чи коректно сервер реагує на помилкові запити: неправильні облікові дані, некоректні параметри моделі, відсутність потрібних даних або звернення до захищених функцій без авторизації.

Першим етапом було перевірено роботу авторизації. Для цього тестувався маршрут входу користувача в систему. У разі передавання коректного логіна і пароля сервер повинен повертати успішну відповідь і дані, необхідні для подальшої роботи клієнта. Якщо користувач вводить неправильні облікові дані, сервер має відхилити запит і повернути повідомлення про помилку.

Наступним етапом було протестовано маршрути роботи з 3D-моделями. Перевірялося отримання списку моделей, створення нової параметричної моделі, перегляд деталей моделі, оновлення параметрів і видалення запису. Особлива увага приділялася створенню моделі з базового примітива, оскільки ця функція є однією з основних у застосунку.

Також було перевірено роботу з шаблонами моделей. Сервер повинен повертати список доступних шаблонів і створювати нову модель на основі вибраної заготовки. Цей сценарій важливий для навчального використання застосунку, оскільки дозволяє користувачу швидко створювати типові моделі без ручного введення всіх параметрів.

Окремо перевірялися маршрути матеріалів і профілів принтерів. Сервер повинен коректно повертати список доступних матеріалів, зокрема PLA, ABS і PETG, а також профілі FDM-принтерів. Ці дані використовуються під час аналізу друкованості та генерації G-code, тому їх коректність впливає на подальші розрахунки.

Важливою частиною тестування була перевірка модуля аналізу друкованості. Для створеної або імпортованої моделі виконувався запит на аналіз, після чого сервер мав повернути розраховані показники: об’єм, площу поверхні, масу

матеріалу, довжину філаменту, орієнтовний час друку та вартість. Під час перевірки оцінювалося, чи повертаються всі необхідні поля та чи не виникає помилка при коректних параметрах моделі.

Маршрути імпорту перевірялися на прикладі STL та OBJ-файлів. Сервер повинен приймати файли підтримуваних форматів, обробляти їх і створювати відповідний запис моделі в системі. Якщо файл має неправильний формат або не може бути оброблений, сервер повинен повернути повідомлення про помилку.

Маршрути експорту перевірялися для форматів STL, OBJ, 3MF та G-code. У разі успішного запиту сервер повинен сформувати файл у вибраному форматі. Для генерації G-code додатково враховуються параметри друку, матеріал і профіль принтера. Така перевірка підтверджує, що backend забезпечує завершальний етап підготовки моделі до FDM-друку.

Таблиця 4.2

Перевірка серверної частини застосунку

Об’єкт тестування	**Що перевірялося**	**Очікуваний результат**
Авторизація	Вхід користувача з коректними та некоректними даними	Сервер допускає користувача лише після правильного входу
Дані користувача	Отримання інформації про поточного користувача	Сервер повертає дані авторизованого користувача
Моделі	Створення, перегляд, редагування та видалення моделей	Дані моделей коректно зберігаються та повертаються клієнту
Шаблони	Отримання шаблонів і створення моделі з шаблону	Нова модель створюється на основі вибраної заготовки
Матеріали	Отримання списку матеріалів	Сервер повертає доступні матеріали для FDM-друку
Профілі принтерів	Отримання характеристик принтерів	Дані профілю використовуються для аналізу і G-code
Аналіз друкованості	Розрахунок об’єму, маси, часу та вартості	Сервер повертає повний набір розрахованих показників
Імпорт	Завантаження STL/OBJ-файлів	Підтримувані файли приймаються, некоректні — відхиляються
Експорт	Формування STL, OBJ, 3MF і G-code	Сервер повертає файл у вибраному форматі
Обробка помилок	Некоректні параметри, відсутність даних, неавторизований доступ	Сервер повертає зрозуміле повідомлення про помилку

Для підтвердження працездатності backend-частини було запущено автоматизовані тести з файлу tests/test_product_api.py. Перевірка охоплювала повний сценарій роботи продукту, захист маршрутів авторизацією, права ролі student, генерацію mesh для розширених примітивів, slicing, CAD-операцію отвору, а також імпортовану OBJ-модель. Результат виконання тестів наведено на рисунку 4.1.

```
$ cd backend && python3 -m pytest -v tests/test_product_api.py
============================= test session starts ==============================
platform linux -- Python 3.13.5, pytest-9.0.2, pluggy-1.6.0 -- /opt/pyvenv/bin/python3
cachedir: .pytest_cache
metadata: {'Python': '3.13.5', 'Platform': 'Linux-4.4.0-x86_64-with-glibc2.41', 'Packages': {'pytest': '9.0.2',
  'pluggy': '1.6.0'}, 'Plugins': {'ddtrace': '4.4.0', 'anyio': '4.13.0', 'cov': '7.0.0', 'asyncio': '1.3.0', 'json-
  report': '1.5.0', 'Faker': '40.1.2', 'metadata': '3.1.1'}, 'PLATFORM': 'linux/amd64', 'CI': 'true'}
rootdir: /mnt/data/app_extract/app_#U0412#U0413-1664/backend
plugins: ddtrace-4.4.0, anyio-4.13.0, cov-7.0.0, asyncio-1.3.0, json-report-1.5.0, Faker-40.1.2, metadata-3.1.1
asyncio: mode=Mode.STRICT, debug=False, asyncio_default_fixture_loop_scope=None,
  asyncio_default_test_loop_scope=function
collecting ... collected 6 items

tests/test_product_api.py::test_full_diploma_product_flow PASSED         [ 16%]
tests/test_product_api.py::test_auth_required_for_model_data_and_exports PASSED [ 33%]
tests/test_product_api.py::test_student_role_can_work_but_cannot_delete_models PASSED [ 50%]
tests/test_product_api.py::test_extended_primitives_generate_mesh_and_slicing PASSED [ 66%]
tests/test_product_api.py::test_mesh_based_slicing_and_cad_hole_feature PASSED [ 83%]
tests/test_product_api.py::test_imported_obj_mesh_can_be_analyzed_exported_and_sliced PASSED [100%]

============================== 6 passed in 3.34s ===============================
```

Рис. 4.1 Результат виконання тестів серверної частини застосунку

За результатами тестування серверної частини встановлено, що backend забезпечує виконання основних функцій застосунку: авторизацію, роботу з моделями, шаблонами, матеріалами, профілями принтерів, аналізом, імпортом, експортом та генерацією G-code. Серверна частина коректно обробляє запити мобільного клієнта, зберігає дані в SQLite та повертає результати у форматі, придатному для відображення в інтерфейсі застосунку.

Таким чином, тестування серверної частини підтвердило її працездатність і відповідність основним функціональним вимогам. Backend-рівень забезпечує необхідну логіку для роботи мобільного клієнта та підтримує базовий сценарій підготовки 3D-моделі до FDM-друку.

4.3 Тест-кейси та результати їх виконання

Після перевірки серверної частини було виконано тестування основних сценаріїв роботи мобільного застосунку. Метою цього етапу було підтвердити, що

користувач може пройти повний базовий шлях підготовки 3D-моделі до FDM-друку: авторизуватися, створити або імпортувати модель, налаштувати параметри друку, виконати аналіз друкованості, переглянути результат та сформувати файл для подальшого використання.

Тестування проводилося з урахуванням функціональних і нефункціональних вимог, визначених у розділі проєктування. Основна увага приділялася не лише факту наявності функцій, а й коректності їх виконання. Для кожного тестового випадку було визначено вхідні дані, очікуваний результат і фактичний результат виконання. Такий підхід дозволяє оцінити працездатність застосунку з позиції реального користувача.

Під час тестування перевірялися такі основні групи функцій: авторизація користувача, робота зі списком моделей, створення параметричної моделі, створення моделі з шаблону, імпорт STL/OBJ-файлів, налаштування параметрів FDM-друку, аналіз друкованості, попередній перегляд, експорт STL/OBJ/3MF та генерація G-code. Окремо перевірялася реакція системи на некоректні дії користувача.

Основні тест-кейси наведено в таблиці 4.3.

Таблиця 4.3

Тест-кейси для перевірки розробленого застосунку

№	Назва тест-кейсу	Вхідні дані / дії користувача	Очікуваний результат	Фактичний результат	Статус
ТС-01	Авторизація з коректними даними	Ввести правильний логін і пароль	Користувач переходить до головного екрана	Вхід виконано, відкрито головний екран	Успішно
ТС-02	Авторизація з некоректними даними	Ввести неправильний пароль	Система показує повідомлення про помилку	Відображено повідомлення про помилку авторизації	Успішно
ТС-03	Завантаження списку моделей	Відкрити екран моделей після входу	Відображається список доступних моделей	Список моделей завантажено	Успішно
ТС-04	Створення параметричної моделі	Обрати примітив “куб”, задати розміри та зберегти	Нова модель з’являється у списку	Модель створено і додано до списку	Успішно
ТС-05	Створення моделі з шаблону	Обрати шаблон і натиснути створення	Система створює модель на основі шаблону	Модель створено з шаблону	Успішно
ТС-06	Імпорт STL-файлу	Завантажити коректний STL-файл	Модель імпортується та зберігається	STL-модель додано до системи	Успішно
ТС-07	Імпорт OBJ-файлу	Завантажити коректний OBJ-файл	Модель імпортується та доступна для аналізу	OBJ-модель імпортовано	Успішно
ТС-08	Імпорт непідтримуваного файлу	Завантажити файл іншого формату	Система відхиляє файл і показує помилку	Відображено повідомлення про непідтримуваний формат	Успішно

Продовження таблиці 4.3

Тест-кейси для перевірки розробленого застосунку

№	Назва тест-кейсу	Вхідні дані / дії користувача	Очікуваний результат	Фактичний результат	Статус
TC-09	Налаштування параметрів друку	Обрати матеріал, висоту шару, заповнення, температури	Параметри зберігаються для моделі	Параметри успішно збережено	Успішно
TC-10	Аналіз друкованості	Запустити аналіз для створеної моделі	Система повертає об’єм, масу, час і вартість	Результати аналізу відображено	Успішно
TC-11	Попередній перегляд моделі	Відкрити preview створеної моделі	Модель відображається у візуальному представленні	Preview сформовано	Успішно
TC-12	Експорт STL	Обрати експорт у STL	Система формує STL-файл	STL-файл сформовано	Успішно
TC-13	Експорт OBJ	Обрати експорт у OBJ	Система формує OBJ-файл	OBJ-файл сформовано	Успішно
TC-14	Експорт 3MF	Обрати експорт у 3MF	Система формує 3MF-файл	3MF-файл сформовано	Успішно
TC-15	Генерація G-code	Обрати генерацію G-code для моделі	Система формує файл керуючих інструкцій	G-code сформовано	Успішно
TC-16	Аналіз моделі з неповними параметрами	Запустити аналіз без обов’язкових даних	Система показує повідомлення про помилку	Відображено повідомлення про недостатність даних	Успішно

Результати тестування показали, що основні функції застосунку працюють відповідно до очікуваної логіки. Користувач може виконати повний базовий сценарій: увійти в систему, створити або імпортувати модель, налаштувати параметри друку, провести аналіз, переглянути модель і сформувати файл у

потрібному форматі. Помилкові ситуації також обробляються коректно: система не завершує роботу аварійно, а повідомляє користувача про проблему.

Окремо було перевірено частину нефункціональних вимог, пов’язаних зі швидкодією, стабільністю, підтримкою форматів і коректністю збереження даних. Результати перевірки наведено в таблиці 4.4.

Таблиця 4.4

Результати перевірки нефункціональних вимог

Вимога	**Критерій перевірки**	**Результат**
Час авторизації	Авторизація має виконуватися не довше ніж за 2 секунди	Виконується в межах встановленого часу
Завантаження списку моделей	Список до 50 моделей має завантажуватися не довше ніж за 3 секунди	Вимогу виконано
Створення параметричної моделі	Створення моделі має виконуватися не довше ніж за 3 секунди	Вимогу виконано
Аналіз друкованості	Аналіз типової навчальної моделі має виконуватися не довше ніж за 5 секунд	Вимогу виконано
Експорт файлів	Експорт STL/OBJ/3MF має виконуватися не довше ніж за 10 секунд	Вимогу виконано
Генерація G-code	Формування G-code для простої моделі має виконуватися не довше ніж за 15 секунд	Вимогу виконано
Підтримка форматів	Імпорт STL/OBJ, експорт STL/OBJ/3MF/G-code	Формати підтримуються
Стабільність роботи	30 послідовних операцій без аварійного завершення	Критичних збоїв не виявлено
Захист доступу	Захищені запити без авторизації мають відхилятися	Неавторизовані запити відхиляються
Обробка помилок	Користувач має отримувати повідомлення про помилку	Повідомлення відображаються коректно

Під час тестування також було перевірено взаємодію між мобільним клієнтом і серверною частиною. Для цього виконувалися сценарії, у яких користувач створює модель у мобільному інтерфейсі, а дані передаються на сервер, зберігаються у базі даних і потім знову відображаються у списку моделей. Така перевірка підтвердила, що клієнтська і серверна частини працюють узгоджено.

За результатами тестування критичних помилок, які унеможливлюють виконання основного сценарію роботи, не виявлено. Реалізований застосунок забезпечує створення параметричних моделей, імпорт файлів, роботу з параметрами друку, аналіз друкованості, експорт моделей і генерацію G-code. Це підтверджує відповідність розробленого програмного продукту основним функціональним вимогам.

Таким чином, тест-кейси та результати їх виконання показали, що мобільний застосунок може використовуватися як навчально-прикладний інструмент для базової підготовки 3D-моделей до FDM-друку. Система виконує основні задачі бакалаврської роботи, коректно обробляє типові дії користувача та забезпечує повний базовий сценарій роботи з моделлю.

ВИСНОВКИ

У результаті виконання бакалаврської роботи було розроблено мобільний застосунок для автоматизації процесу моделювання під час 3D-друку. Розроблений програмний продукт орієнтований на студентів, початківців у сфері 3D-друку та користувачів навчальних FDM-принтерів, яким необхідний зрозумілий інструмент для створення простих 3D-моделей, налаштування параметрів друку, аналізу друкованості та формування файлів для подальшого використання.

У першому розділі було проаналізовано предметну галузь 3D-друку, особливості процесу моделювання та підготовки моделей до друку. Було визначено, що підготовка 3D-моделі до FDM-друку включає не лише створення геометрії, а й перевірку параметрів, вибір матеріалу, налаштування технологічних характеристик, аналіз друкованості, експорт моделі та формування G-code. Також було розглянуто основні формати, які використовуються у процесі 3D-друку: STL, OBJ, 3MF та G-code.

У ході аналізу існуючих програмних засобів було розглянуто Tinkercad, Autodesk Fusion, FreeCAD, Blender, UltiMaker Cura та PrusaSlicer. Встановлено, що кожен із цих інструментів ефективно виконує окрему групу задач, однак для початківців процес часто залишається розділеним між кількома програмами. Одні засоби краще підходять для моделювання, інші — для slicing та генерації G-code. Саме тому було обґрунтовано актуальність розробки мобільного застосунку, який поєднує базове параметричне моделювання, аналіз друкованості, експорт моделей і генерацію G-code в одному навчально-прикладному середовищі.

У другому розділі було виконано проєктування мобільного застосунку. Було визначено загальну характеристику програмного продукту, його клієнт-серверну архітектуру, функціональні й нефункціональні вимоги, сценарії використання, структуру бази даних та інтерфейс користувача. Для системи було обрано архітектуру, у якій мобільний клієнт відповідає за взаємодію з користувачем, а серверна частина — за обробку даних, роботу з моделями, аналіз, імпорт, експорт

та генерацію G-code. Такий підхід дозволив логічно розділити відповідальність між компонентами системи.

У третьому розділі було описано реалізацію програмного продукту. Мобільний клієнт реалізовано з використанням Flutter та мови Dart. Серверну частину створено мовою Python із використанням FastAPI. Для збереження даних використано SQLite. У застосунку реалізовано авторизацію користувачів, роботу з моделями, шаблонами, матеріалами, профілями принтерів, параметрами друку, імпортом STL/OBJ-файлів, аналізом друкованості, експортом у формати STL, OBJ, 3MF та генерацією G-code.

Практична новизна роботи полягає у поєднанні в одному мобільному навчально-прикладному застосунку функцій, які зазвичай розподілені між кількома окремими програмними засобами. Розроблений застосунок дозволяє користувачу створити просту параметричну модель, використати шаблон, імпортувати готовий файл, налаштувати параметри FDM-друку, виконати базовий аналіз друкованості, переглянути модель і сформувати результат у потрібному форматі. Така комбінація функцій є корисною саме для початкового та навчального рівня роботи з 3D-друком.

У четвертому розділі було проведено тестування програмного продукту. Було обґрунтовано вибір видів тестування, перевірено серверну частину застосунку, а також сформовано тест-кейси для основних сценаріїв роботи. Під час тестування перевірялися авторизація, створення параметричних моделей, створення моделей із шаблонів, імпорт STL/OBJ-файлів, налаштування параметрів друку, аналіз друкованості, попередній перегляд, експорт моделей та генерація G-code. За результатами перевірки встановлено, що застосунок виконує основні функції відповідно до поставлених вимог.

Мета бакалаврської роботи була досягнута: процес створення та підготовки 3D-моделей до FDM-друку було спрощено за рахунок розробки мобільного застосунку з функціями параметричного моделювання, аналізу друкованості, налаштування параметрів друку, попереднього перегляду, експорту моделей і генерації G-code.

Розроблений програмний продукт може бути використаний як навчально-прикладний інструмент для ознайомлення з основами параметричного 3D-моделювання та підготовки моделей до FDM-друку. Він не є промисловою CAD-системою або повноцінним професійним slicer-рішенням, однак забезпечує достатню функціональність для базового сценарію роботи з 3D-моделлю. Надалі застосунок може бути розширений підтримкою складніших типів моделей, більш точним аналізом геометрії, додатковими матеріалами, профілями принтерів та покращеним модулем slicing.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Задніченко Д.А., Щербина І.С. СУЧАСНІ МЕТОДИ АВТОМАТИЗАЦІЇ 3D-ДРУКУ. Матеріали ІІ Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій.
Збірник тез. – К.: ДУІКТ, 2025, С.450 -452.

2. Задніченко Д.А., Щербина І.С. АВТОМАТИЗАЦІЯ ПРОЦЕСУ МОДЕЛЮВАННЯ 3D ДРУКУ ШЛЯХОМ РОЗРОБКИ МОБІЛЬНОГО КРОСПЛАТФОРМЕННОГО ЗАСТОСУНКУ ОНЛАЙН-КІНОТЕАТРУ.
Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях»,
23 квітня 2026 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2026, С.338-340.

ПЕРЕЛІК ПОСИЛАНЬ

1. ISO/ASTM 52900:2021(en). Additive manufacturing — General principles — Fundamentals and vocabulary [Електронний ресурс]. URL: https://www.iso.org/obp/ui/#iso:std:iso-astm:52900:ed-2:v1:en (дата звернення: 05.05.2026).

2. NIST. Additive Manufacturing [Електронний ресурс]. National Institute of Standards and Technology. URL: https://www.nist.gov/additive-manufacturing (дата звернення: 05.05.2026).

3. Cano-Vicent A., Tambuwala M. M., Hassan S. S., Barh D., Aljabali A. A. A., Birkett M., Arjunan A., Serrano-Aroca Á. Fused deposition modelling: Current status, methodology, applications and future prospects. Additive Manufacturing. 2021. Vol. 47. Article 102378. DOI: 10.1016/j.addma.2021.102378 [Електронний ресурс]. URL: https://doi.org/10.1016/j.addma.2021.102378 (дата звернення: 05.05.2026).

4. Hüner B., et al. An Overview of Various Additive Manufacturing Technologies and Materials for Electrochemical Applications. Micromachines. 2022. Vol. 13, Iss. 11. [Електронний ресурс]. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC9670698/ (дата звернення: 05.05.2026).

5. Гусєв В. Аналіз файлів у форматі STL, як основа моделювання у задачах адитивного виробництва [Електронний ресурс]. Технічні науки та технології. 2024. URL: https://tst.stu.cn.ua/article/view/302673 (дата звернення: 05.05.2026).

6. Library of Congress. STL (STereoLithography) File Format Family [Електронний ресурс]. Sustainability of Digital Formats. URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml (дата звернення: 05.05.2026).

7. Library of Congress. Wavefront OBJ File Format [Електронний ресурс]. Sustainability of Digital Formats. URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml (дата звернення: 05.05.2026).

8. 3MF Consortium. 3D Manufacturing Format Specification [Електронний ресурс]. URL: https://3mf.io/spec/ (дата звернення: 05.05.2026).
9. Marlin Firmware. G-code Commands [Електронний ресурс]. URL: https://marlinfw.org/meta/gcode/ (дата звернення: 05.05.2026).
10. Autodesk. Tinkercad: 3D design, electronics and coding [Електронний ресурс]. URL: https://www.tinkercad.com/ (дата звернення: 05.05.2026).
11. Autodesk. Autodesk Fusion: 3D CAD, CAM, CAE, PCB and PDM software [Електронний ресурс]. URL: https://www.autodesk.com/products/fusion-360/overview (дата звернення: 05.05.2026).
12. FreeCAD. FreeCAD: Your own 3D parametric modeler [Електронний ресурс]. URL: https://www.freecad.org/ (дата звернення: 05.05.2026).
13. Blender Foundation. Blender — Free and Open Source 3D Creation Software [Електронний ресурс]. URL: https://www.blender.org/ (дата звернення: 05.05.2026).
14. UltiMaker. UltiMaker Cura: 3D printing software [Електронний ресурс]. URL: https://ultimaker.com/software/ultimaker-cura/ (дата звернення: 05.05.2026).
15. Prusa Research. PrusaSlicer [Електронний ресурс]. URL: https://www.prusa3d.com/p/prusaslicer/ (дата звернення: 05.05.2026).
16. Flutter. Flutter documentation [Електронний ресурс]. URL: https://docs.flutter.dev/ (дата звернення: 05.05.2026).
17. Dart. Dart documentation [Електронний ресурс]. URL: https://dart.dev/docs (дата звернення: 05.05.2026).
18. Python Software Foundation. Python documentation [Електронний ресурс]. URL: https://docs.python.org/3/ (дата звернення: 05.05.2026).
19. FastAPI. FastAPI documentation [Електронний ресурс]. URL: https://fastapi.tiangolo.com/ (дата звернення: 05.05.2026).
20. SQLite Consortium. SQLite documentation [Електронний ресурс]. URL: https://sqlite.org/docs.html (дата звернення: 05.05.2026).
21. MDN Web Docs. HTTP overview [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview (дата звернення: 05.05.2026).

ДОДАТОК А ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ПРЕЗЕНТАЦІЯ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мобільний застосунок для автоматизації процесу моделювання під час 3D друку

Виконав студент 4 курсу
групи ПД-42
Задніченко Дмитро Андрійович
Керівник роботи
доц. кафедри, канд. тех. наук, доц. Щербина Ірина Сергіївна

Київ – 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи : спрощення процесу створення та підготовки 3D-моделей до FDM-друку за рахунок мобільного застосунку з функціями параметричного моделювання, аналізу друкованості, експорту моделей та генерації G-code.

Об'єкт дослідження : процес моделювання та підготовки 3D-моделей до FDM-друку.

Предмет дослідження : програмні засоби автоматизації базового параметричного
3D-моделювання, аналізу друкованості та підготовки моделей до друку в мобільному застосунку.

АНАЛІЗ АНАЛОГІВ

	Tinkercad	Autodesk Fusion	FreeCAD	Cura	Model3D Print
Параметричне створення простих моделей	+	+	+	-	+
Підготовка саме до FDM-друку	-	+	+	+	+
Аналіз друкованості та розрахунок показників	-	+	-	+	+
Експорт STL / OBJ / 3MF	-	+	+	+	+
Генерація G-code	-	+	-	+	+
Єдиний навчальний мобільний сценарій	-	-	-	-	+

4

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати предметну галузь 3D-друку та етапи підготовки моделей до FDM-друку.
2. Розглянути існуючі програмні засоби для 3D-моделювання та підготовки до друку.
3. Сформувати функціональні та нефункціональні вимоги до мобільного застосунку.
4. Спроєктувати архітектуру системи, структуру бази даних, сценарії використання та інтерфейс.
5. Реалізувати серверну частину на FastAPI та мобільний клієнт на Flutter/Dart.
6. Реалізувати створення параметричних моделей, імпорт, аналіз друкованості, експорт і генерацію G-code.
7. Провести тестування програмного продукту та перевірити відповідність реалізованих функцій вимогам.

3

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги:
1. Авторизація користувача та розмежування ролей.
2. Створення параметричних моделей: куб, циліндр, сфера, конус, піраміда, тор.
3. Використання готових шаблонів моделей.
4. Імпорт моделей у форматах STL та OBJ.
5. Виконання операцій над моделлю: масштабування, поворот, отвір, нахил, дзеркальне відображення.
6. Налаштування матеріалу, профілю принтера і параметрів друку.
7. Експорт STL / OBJ / 3MF і генерація G-code.

Нефункціональні вимоги:
1. Основні дії виконуються не більше ніж за 3 натискання.
2. Відповідь API має надходити не довше ніж за 2 секунди.
3. Список до 100 проєктів має відкриватися не довше ніж за 1 секунду.
4. Повідомлення про помилку має з'являтися не пізніше ніж за 1 секунду; підтримка файлів до 50 МБ.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Dart

Flutter

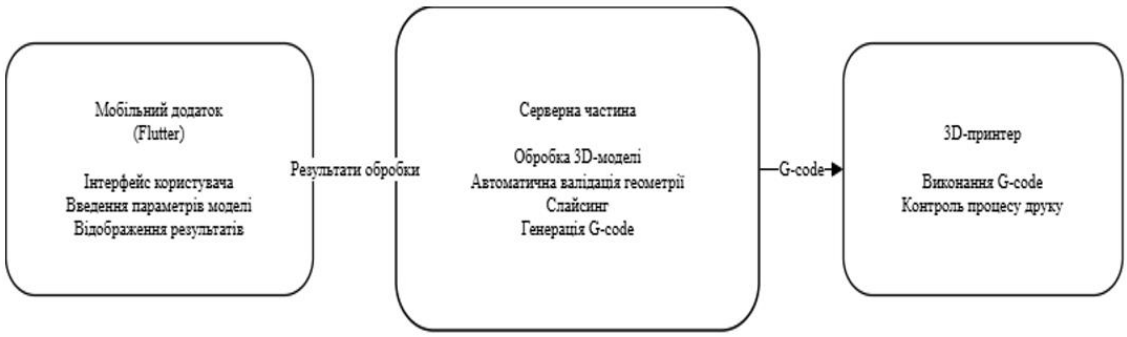
Python

6

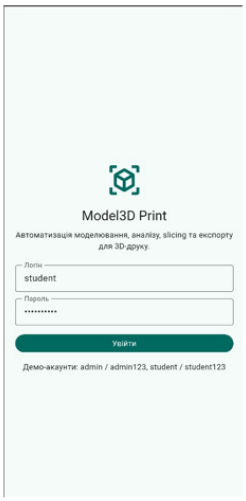
ДІАГРАМА ОСНОВНИХ ЕТАПІВ ПРОЦЕСУ 3D-друку

Рисунок 1.3 – Основні етапи процесу 3D-друку

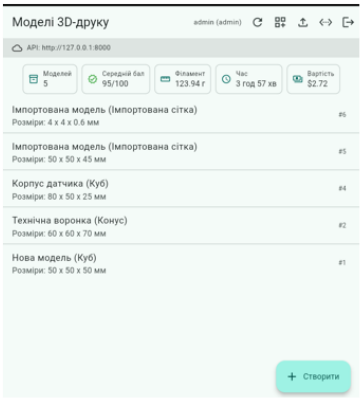
Клієнт–серверна архітектура програмних систем 3D-друку



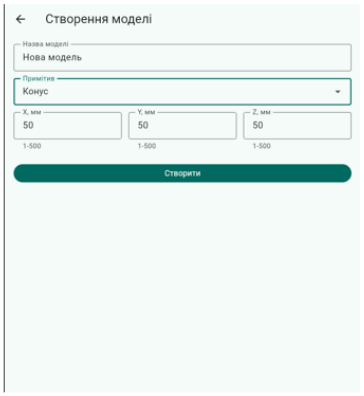
ЕКРАННІ ФОРМИ



Авторизація користувача

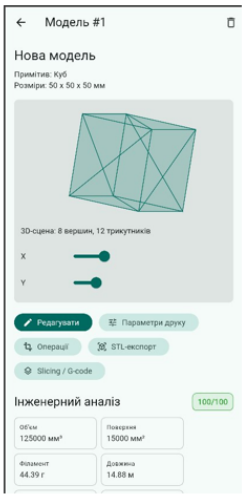


Список моделей

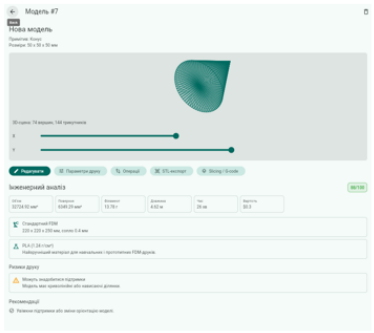


Створення параметричної моделі

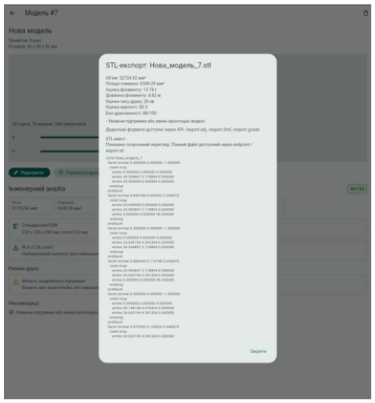
ЕКРАННІ ФОРМИ



Деталі моделі



3D-preview і параметри



Перегляд результату експорту

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1.Задніченко Д.А., Щербина І.С. Сучасні методи автоматизації 3D-Друку Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.450 -452.

2 Задніченко Д.А., Щербина І.С. Автоматизація процесу моделювання 3D друку шляхом розробки мобільного застосунку. Матеріали Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. *(подано до друку).*

11

ВИСНОВКИ

1. Проаналізовано предметну галузь FDM-друку, етапи підготовки 3D-моделей та існуючі програмні засоби.
2. Сформовано функціональні й нефункціональні вимоги до навчально-прикладного мобільного застосунку.
3. Спроєктовано клієнт-серверну архітектуру, структуру бази даних, сценарії використання та інтерфейс користувача.
4. Реалізовано backend на FastAPI, мобільний клієнт на Flutter/Dart і збереження даних у SQLite.
5. Реалізовано створення параметричних моделей, роботу з шаблонами, імпорт STL/OBJ, аналіз друкованості, експорт STL/OBJ/3MF та генерацію G-code.
6. Проведено тестування ключових сценаріїв, що підтвердило відповідність програмного продукту поставленим вимогам.

12

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНОГО МОДУЛЮ

Лістинг А.1 – Ініціалізація FastAPI, основні типи даних та моделі запитів

```
# backend/main.py
# Основні фрагменти серверної частини застосунку

from fastapi import Depends, FastAPI, Header,
HTTPException
from fastapi.middleware.cors import CORSMiddleware
from fastapi.responses import Response
from pydantic import BaseModel, ConfigDict, Field
import hashlib
import json
import math
import secrets
import sqlite3
from datetime import datetime, timezone
from pathlib import Path
from typing import Literal

        BASE_DIR = Path(__file__).resolve().parent
        DB_PATH = BASE_DIR / "app.db"

        app = FastAPI(
            title="3D Model Automation API",
            version="2.0.0",
            description=(
                "API для параметричного 3D-моделювання,
імпорту STL/OBJ, "
                "CAD-операцій, mesh-based slicing та
експорту файлів для FDM-друку."
            ),
        )

        app.add_middleware(
            CORSMiddleware,
            allow_origins=["*"],
            allow_credentials=False,
            allow_methods=["*"],
            allow_headers=["*"],
        )

        Primitive = Literal["Куб", "Циліндр", "Сфера",
"Конус", "Піраміда", "Тор", "Імпортована сітка"]
        Material = Literal["PLA", "ABS", "PETG"]
        Role = Literal["admin", "student"]
        OperationType = Literal["scale", "translate", "mirror",
"rotate", "taper", "twist", "skew", "hole"]

class AppModel(BaseModel):
            model_config                                =
ConfigDict(str_strip_whitespace=True)

class ModelCreate(AppModel):
            name: str = Field(min_length=1, max_length=80)
            primitive: Primitive
            size_x: float = Field(gt=0, le=500)
            size_y: float = Field(gt=0, le=500)
            size_z: float = Field(gt=0, le=500)

class PrintParams(AppModel):
            material: Material = "PLA"
            layer_height: float = Field(default=0.2, ge=0.05,
le=0.4)
            infill_density: int = Field(default=20, ge=0, le=100)
            print_speed: int = Field(default=60, ge=10, le=180)
            nozzle_temperature: int | None =
Field(default=None, ge=150, le=280)
            bed_temperature: int | None = Field(default=None,
ge=0, le=120)
            supports: bool = False
            printer_profile: str = "Стандартний FDM"
            nozzle_diameter: float = Field(default=0.4, gt=0,
le=1.2)
            filament_price_per_kg: float = Field(default=22.0,
gt=0, le=500)
            quality: str = "Стандарт"

class ModelOperationCreate(AppModel):
            type: OperationType
            axis: Literal["x", "y", "z"] | None = None
            value: float
            value_y: float | None = None
            value_z: float | None = None
            note: str = ""

class UserOut(AppModel):
            id: int
            username: str
            role: Role

class ModelAnalysis(AppModel):
            volume_cm3: float
            surface_area_cm2: float
            mass_g: float
            filament_length_m: float
            estimated_time_min: int
            estimated_cost: float
            printability_score: int
            issues: list[dict]
```

Лістинг А.2 – Ініціалізація бази даних SQLite та авторизація користувача

```
def connection() -> sqlite3.Connection:
            DB_PATH.parent.mkdir(parents=True,
exist_ok=True)
            con = sqlite3.connect(DB_PATH)
            con.execute("PRAGMA foreign_keys = ON;")
    return con


def hash_password(password: str, salt: str) -> str:
    return hashlib.sha256(f"{salt}:{password}".encode("utf-
8")).hexdigest()


def init_db() -> None:
            with connection() as con:
                cur = con.cursor()
                cur.execute("""
                    CREATE TABLE IF NOT EXISTS models(
                        id      INTEGER      PRIMARY      KEY
AUTOINCREMENT,
                        name TEXT NOT NULL,
```

```
                primitive TEXT NOT NULL,
                size_x REAL NOT NULL,
                size_y REAL NOT NULL,
                size_z REAL NOT NULL,
                created_at TEXT NOT NULL
            );
        """)
        cur.execute("""
            CREATE TABLE IF NOT EXISTS print_params(
                model_id INTEGER PRIMARY KEY,
                material TEXT NOT NULL DEFAULT 'PLA',
                layer_height REAL NOT NULL DEFAULT 0.2,
                infill_density INTEGER NOT NULL DEFAULT 20,
                print_speed INTEGER NOT NULL DEFAULT 60,
                nozzle_temperature INTEGER,
                bed_temperature INTEGER,
                supports INTEGER NOT NULL DEFAULT 0,
                printer_profile TEXT NOT NULL DEFAULT 'Стандартний FDM',
                nozzle_diameter REAL NOT NULL DEFAULT 0.4,
                filament_price_per_kg REAL NOT NULL DEFAULT 22.0,
                quality TEXT NOT NULL DEFAULT 'Стандарт',
                FOREIGN KEY(model_id) REFERENCES models(id) ON DELETE CASCADE
            );
        """)
        cur.execute("""
            CREATE TABLE IF NOT EXISTS model_operations(
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                model_id INTEGER NOT NULL,
                type TEXT NOT NULL,
                axis TEXT,
                value REAL NOT NULL,
                value_y REAL,
                value_z REAL,
                note TEXT NOT NULL DEFAULT '',
                created_at TEXT NOT NULL,
                FOREIGN KEY(model_id) REFERENCES models(id) ON DELETE CASCADE
            );
        """)
        cur.execute("""
            CREATE TABLE IF NOT EXISTS model_meshes(
                model_id INTEGER PRIMARY KEY,
                source_format TEXT NOT NULL,
                source_name TEXT NOT NULL,
                vertices_json TEXT NOT NULL,
                faces_json TEXT NOT NULL,
                FOREIGN KEY(model_id) REFERENCES models(id) ON DELETE CASCADE
            );
        """)
        con.commit()


def create_session(user_id: int) -> str:
    token = secrets.token_urlsafe(32)
    with connection() as con:
        con.execute(
            "INSERT INTO sessions(token,user_id,created_at) VALUES(?,?,?);",
            (token, user_id, datetime.now(timezone.utc).isoformat()),
        )
        con.commit()
    return token


def current_user(authorization: str | None = Header(default=None)) -> UserOut:
    if not authorization or not authorization.startswith("Bearer "):
        raise HTTPException(status_code=401, detail="Потрібна авторизація")
    token = authorization.removeprefix("Bearer ").strip()
    with connection() as con:
        row = con.execute("""
            SELECT users.id, users.username, users.role
            FROM sessions
            JOIN users ON users.id = sessions.user_id
            WHERE sessions.token = ?;
        """, (token,)).fetchone()
    if not row:
        raise HTTPException(status_code=401, detail="Сесія недійсна")
    return UserOut(id=row[0], username=row[1], role=row[2])
```

Лістинг А.3 – Формування геометрії моделі та застосування операцій

```
def build_triangles(model) -> list:
    hole_operations = [op for op in model.operations if op.type == "hole"]
    transform_operations = [op for op in model.operations if op.type != "hole"]

    if model.primitive == "Імпортована сітка":
        vertices, faces, _, _ = get_imported_mesh(model.id)
        triangles = triangles_from_mesh(vertices, faces)
    elif model.primitive == "Куб":
        triangles = box_triangles(model.size_x, model.size_y, model.size_z)
    elif model.primitive == "Циліндр":
        triangles = cylinder_triangles(model.size_x, model.size_y, model.size_z)
    elif model.primitive == "Сфера":
        triangles = sphere_triangles(model.size_x, model.size_y, model.size_z)
    elif model.primitive == "Конус":
        triangles = cone_triangles(model.size_x, model.size_y, model.size_z)
    elif model.primitive == "Піраміда":
        triangles = pyramid_triangles(model.size_x, model.size_y, model.size_z)
    elif model.primitive == "Тор":
        triangles = torus_triangles(model.size_x, model.size_y, model.size_z)
    else:
        raise HTTPException(status_code=400, detail="Непідтримуваний примітив")

    if hole_operations and model.primitive == "Куб":
        triangles = box_with_hole_triangles(
            model.size_x, model.size_y, model.size_z, hole_operations[-1].value
        )
    return apply_operations(triangles, transform_operations)
```

```
def transform_vertex(vertex, operation, bounds=None):
        x, y, z = vertex
        min_x, max_x, min_y, max_y, min_z, max_z = bounds or (0, 1, 0, 1, 0, 1)
        height_factor = (z - min_z) / max(max_z - min_z, 1e-6)

        if operation.type == "scale":
            sx = operation.value
            sy = operation.value_y if operation.value_y is not None else operation.value
            sz = operation.value_z if operation.value_z is not None else operation.value
            return (x * sx, y * sy, z * sz)
        if operation.type == "translate":
            return (x + operation.value, y + (operation.value_y or 0), z + (operation.value_z or 0))
        if operation.type == "rotate":
            angle = math.radians(operation.value)
            cos_a, sin_a = math.cos(angle), math.sin(angle)
            if operation.axis == "z":
                return (x * cos_a - y * sin_a, x * sin_a + y * cos_a, z)
        if operation.type == "taper":
            factor = 1 + (operation.value - 1) * height_factor
            return (x * factor, y * factor, z)
        if operation.type == "twist":
            angle = math.radians(operation.value * height_factor)
            return (x * math.cos(angle) - y * math.sin(angle), x * math.sin(angle) + y * math.cos(angle), z)
    return vertex


def apply_operations(triangles: list, operations: list) -> list:
        transformed = triangles
        for operation in operations:
            bounds = triangle_bounds(transformed)
            transformed = [
                (
                    transform_vertex(tri[0], operation, bounds),
                    transform_vertex(tri[1], operation, bounds),
                    transform_vertex(tri[2], operation, bounds),
                )
                for tri in transformed
            ]
    return transformed
```

Лістинг А.4 – Аналіз друкованості, експорт STL та генерація G-code

```
def estimate_print(model, surface_area: float) -> dict:
        params = active_print_params(model)
        volume_cm3 = model_volume(model)
        density = material_density(params.material)
        mass_g = volume_cm3 * density * (0.18 + params.infill_density / 100 * 0.82)
        filament_length_m = mass_g / max(density, 0.01) / 2.405
        time_min = int(max(5, surface_area * 0.7 / max(params.print_speed, 1) + volume_cm3 * 2.5))
        cost = mass_g / 1000 * params.filament_price_per_kg
    return {
            "mass_g": round(mass_g, 2),
            "filament_length_m": round(filament_length_m, 2),
            "estimated_time_min": time_min,
            "estimated_cost": round(cost, 2),
        }
def analyze_model_payload(model) -> ModelAnalysis:
        triangles = build_triangles(model)
        surface_area = sum(triangle_area(tri) for tri in triangles) / 100
        bounds = triangle_bounds(triangles)
        x_span = bounds[1] - bounds[0]
        y_span = bounds[3] - bounds[2]
        z_span = bounds[5] - bounds[4]
        estimate = estimate_print(model, surface_area)
        issues = []
        score = 100

        if max(x_span, y_span, z_span) > 220:
            issues.append({"severity": "danger", "message": "Модель може перевищувати область друку"})
            score -= 25
        if min(x_span, y_span, z_span) < 2:
            issues.append({"severity": "warning", "message": "Окремі елементи можуть бути занадто тонкими"})
            score -= 10
        if active_print_params(model).supports is False and z_span > 80:
            issues.append({"severity": "warning", "message": "Для високих або складних моделей можуть знадобитися підтримки"})
            score -= 10

    return ModelAnalysis(
            volume_cm3=round(mesh_volume(triangles) / 1000, 2),
            surface_area_cm2=round(surface_area, 2),
            printability_score=max(0, min(100, score)),
            issues=issues,
            **estimate,
        )


@app.post("/models", response_model=ModelCreate)
def create_model(data: ModelCreate, user: UserOut = Depends(current_user)):
        created_at = datetime.now(timezone.utc).isoformat()
        with connection() as con:
            cur = con.cursor()
            cur.execute(
                "INSERT INTO models(name,primitive,size_x,size_y,size_z,created_at) VALUES(?,?,?,?,?,?);",
                (data.name, data.primitive, data.size_x, data.size_y, data.size_z, created_at),
            )
            con.commit()
    return data


@app.post("/models/import")
def import_mesh_model(payload, user: UserOut = Depends(current_user)):
        vertices, faces, dimensions = parse_imported_mesh(payload)
        model = create_model(ModelCreate(
            name=payload.name,
            primitive="Імпортована сітка",
            size_x=max(dimensions[0], 0.01),
            size_y=max(dimensions[1], 0.01),
            size_z=max(dimensions[2], 0.01),
        ), user)
    # vertices/faces зберігаються у таблиці model_meshes у форматі JSON.
```

```
    return {"model": model, "vertex_count": len(vertices),
"face_count": len(faces)}


@app.get("/models/{model_id}/analysis",
response_model=ModelAnalysis)
def analyze_model(model_id: int, user: UserOut =
Depends(current_user)) -> ModelAnalysis:
    return
analyze_model_payload(get_model_or_404(model_id))


@app.get("/models/{model_id}/export.stl")
```

Лістинг А.5 – Ініціалізація Flutter-застосунку та основні моделі даних

```
// mobile/lib/main.dart
// Основні фрагменти мобільного клієнта Flutter

import 'dart:convert';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: '3D Model Automation',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(useMaterial3: true,
colorSchemeSeed: Colors.indigo),
      home: const LoginPage(),
    );
  }
}

class ModelItem {
  final int id;
  final String name;
  final String primitive;
  final double sizeX;
  final double sizeY;
  final double sizeZ;

  ModelItem({
    required this.id,
    required this.name,
    required this.primitive,
    required this.sizeX,
    required this.sizeY,
    required this.sizeZ,
  });

  factory ModelItem.fromJson(Map<String, dynamic>
json) => ModelItem(
    id: json['id'],
    name: json['name'],
    primitive: json['primitive'],
    sizeX: (json['size_x'] as num).toDouble(),
    sizeY: (json['size_y'] as num).toDouble(),
    sizeZ: (json['size_z'] as num).toDouble(),
  );

  Map<String, dynamic> toJson() => {
    'name': name,
    'primitive': primitive,
    'size_x': sizeX,
    'size_y': sizeY,
    'size_z': sizeZ,
  };
}

class PrintParams {
  final String material;
  final double layerHeight;
  final int infillDensity;
  final int printSpeed;
  final bool supports;
  final String printerProfile;

  const PrintParams({
    this.material = 'PLA',
    this.layerHeight = 0.2,
    this.infillDensity = 20,
    this.printSpeed = 60,
    this.supports = false,
    this.printerProfile = 'Стандартний FDM',
  });

  Map<String, dynamic> toJson() => {
    'material': material,
    'layer_height': layerHeight,
    'infill_density': infillDensity,
    'print_speed': printSpeed,
    'supports': supports,
    'printer_profile': printerProfile,
  };
}
```

Лістинг А.6 – API-клієнт мобільного застосунку

```
class Api {
  static const String baseUrl = 'http://127.0.0.1:8000';
  String? token;

  Map<String, String> get headers => {
    'Content-Type': 'application/json',
    if (token != null) 'Authorization': 'Bearer $token',
  };

  Future<void> login(String username, String
password) async {
    final response = await http.post(
      Uri.parse('$baseUrl/auth/login'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode({'username': username,
'password': password}),
    );
    if (response.statusCode != 200) {
      throw Exception('Помилка авторизації');
    }
    token = jsonDecode(response.body)['token'];
  }
```

```
def export_model_stl(model_id: int, user: UserOut =
Depends(current_user)) -> Response:
    model = get_model_or_404(model_id)
    content = render_ascii_stl(model.name,
build_triangles(model))
    return Response(content=content, media_type="model/stl")


@app.get("/models/{model_id}/export.gcode")
def export_model_gcode(model_id: int, user: UserOut =
Depends(current_user)) -> Response:
    gcode = generate_gcode(model_id).gcode
    return Response(content=gcode, media_type="text/x.gcode")
```

```
  Future<List<ModelItem>> fetchModels() async {
    final response = await http.get(Uri.parse('$baseUrl/models'), headers: headers);
    if (response.statusCode != 200) throw Exception('Не вдалося завантажити моделі');
    return (jsonDecode(response.body) as List)
        .map((item) => ModelItem.fromJson(item))
        .toList();
  }

  Future<ModelItem> createModel(ModelItem model) async {
    final response = await http.post(
      Uri.parse('$baseUrl/models'),
      headers: headers,
      body: jsonEncode(model.toJson()),
    );
    if (response.statusCode != 200) throw Exception('Не вдалося створити модель');
    return ModelItem.fromJson(jsonDecode(response.body));
  }

  Future<Map<String, dynamic>> analyzeModel(int id) async {
    final response = await http.get(Uri.parse('$baseUrl/models/$id/analysis'), headers: headers);
    if (response.statusCode != 200) throw Exception('Не вдалося виконати аналіз');
    return jsonDecode(response.body);
  }

  Future<String> exportStl(int id) async {
    final response = await http.get(Uri.parse('$baseUrl/models/$id/export.stl'), headers: headers);
    if (response.statusCode != 200) throw Exception('Не вдалося експортувати STL');
    return response.body;
  }

  Future<String> exportGcode(int id) async {
    final response = await http.get(Uri.parse('$baseUrl/models/$id/export.gcode'), headers: headers);
    if (response.statusCode != 200) throw Exception('Не вдалося сформувати G-code');
    return response.body;
  }
}
```

Лістинг А.7 – Основні екрани авторизації, списку моделей та перегляду моделі

```
class LoginPage extends StatefulWidget {
  const LoginPage({super.key});
  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final loginController = TextEditingController(text: 'student');
  final passwordController = TextEditingController(text: 'student123');
  final api = Api();

  Future<void> submit() async {
    await api.login(loginController.text, passwordController.text);
    if (!mounted) return;
    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (_) => ModelsPage(api: api)),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Card(
          child: Padding(
            padding: const EdgeInsets.all(24),
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                const Text('3D Model Automation', style: TextStyle(fontSize: 24)),
                TextField(controller: loginController, decoration: const InputDecoration(labelText: 'Логін')),
                TextField(controller: passwordController, decoration: const InputDecoration(labelText: 'Пароль'), obscureText: true),
                const SizedBox(height: 16),
                FilledButton(onPressed: submit, child: const Text('Увійти')),
              ],
            ),
          ),
        ),
      ),
    );
  }
}

class ModelsPage extends StatefulWidget {
  final Api api;
  const ModelsPage({super.key, required this.api});
  @override
  State<ModelsPage> createState() => _ModelsPageState();
}

class _ModelsPageState extends State<ModelsPage> {
  late Future<List<ModelItem>> modelsFuture;

  @override
  void initState() {
    super.initState();
    modelsFuture = widget.api.fetchModels();
  }

  void openDetails(ModelItem model) {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (_) => ModelDetailsPage(api: widget.api, model: model)),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Мої 3D-моделі')),
      body: FutureBuilder<List<ModelItem>>(
        future: modelsFuture,
        builder: (context, snapshot) {
```

```
                if (!snapshot.hasData) return const Center(child:
CircularProgressIndicator());
                final models = snapshot.data!;
        return ListView.builder(
                  itemCount: models.length,
                  itemBuilder: (_, index) => ListTile(
                    title: Text(models[index].name),
                    subtitle:  Text('${models[index].primitive}  •
${models[index].sizeX}x${models[index].sizeY}x${models[in
dex].sizeZ} мм'),
                    onTap: () => openDetails(models[index]),
                  ),
                );
              },
            ),
          );
        }
      }

class ModelDetailsPage extends StatelessWidget {
        final Api api;
        final ModelItem model;
        const      ModelDetailsPage({super.key,      required
this.api, required this.model});

        Future<void>  showAnalysis(BuildContext  context)
async {
          final analysis = await api.analyzeModel(model.id);
          if (!context.mounted) return;
          showDialog(
            context: context,
            builder: (_) => AlertDialog(
              title: const Text('Аналіз друкованості'),
              content: Text(
                'Об’єм: ${analysis['volume_cm3']} см³\n'
                'Маса: ${analysis['mass_g']} г\n'
                'Час: ${analysis['estimated_time_min']} хв\n'
                'Оцінка: ${analysis['printability_score']} / 100',
              ),
            ),
          );
        }

        Future<void>    showGcode(BuildContext    context)
async {
          final gcode = await api.exportGcode(model.id);
          if (!context.mounted) return;
          showDialog(
            context: context,
            builder: (_) => AlertDialog(
              title: const Text('G-code'),
              content:              SingleChildScrollView(child:
Text(gcode.substring(0, gcode.length.clamp(0, 2000)))),
            ),
          );
        }

  @override
        Widget build(BuildContext context) {
    return Scaffold(
            appBar: AppBar(title: Text(model.name)),
            body: Padding(
              padding: const EdgeInsets.all(16),
              child: Column(
                crossAxisAlignment:
CrossAxisAlignment.stretch,
                children: [
                  Text('Тип: ${model.primitive}'),
                  Text('Розміри:        ${model.sizeX}        x
${model.sizeY} x ${model.sizeZ} мм'),
                  const SizedBox(height: 16),
                  FilledButton(onPressed:        ()        =>
showAnalysis(context),      child:      const      Text('Аналіз
друкованості')),
                  OutlinedButton(onPressed:        ()        =>
showGcode(context), child: const Text('Згенерувати G-code')),
                ],
              ),
            ),
          );
        }
      }
```