**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

Навчально-науковий інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

# Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ОСВІТНЬОГО ПОРТАЛУ З НАВЧАННЯ ПРОГРАМУВАННЮ, ВИКОРИСТОВУЮЧИ ПАТЕРН ПРОЕКТУВАННЯ MVC ТА ПРИНЦИПІВ SOLID»**

Виконав: студент 5 курсу, групи ППЗ-52
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Рудник К.О.
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2021

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**Навчально-науковий інститут Телекомунікацій**

Кафедра Телекомунікаційних технологій
Ступінь вищої освіти - «Бакалавр»
Напрям підготовки -121 Інженерія програмного забезпечення

**ЗАТВЕРДЖУЮ**
Завідувач кафедри
Телекомунікаційних технологій
А.В. Бондаренко
—__‖_____2021 року

# З А В Д А Н Н Я
## НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

**Рудник Костянтин Олегович**
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка освітнього порталу з навчання програмуванню, використовуючи патерн проектування mvc та принципів solid»

Керівник роботи Гаманюк І.М.
затверджені наказом вищого навчального закладу від —_‖_____2021 року №__.
2. Строк подання студентом роботи
3. Вхідні дані до роботи:
Шаблони проектування та
Model-View-Controller
Реалізація принципів SOLID
Науково-технічна література з питань, пов'язаних з побудовою освітнього порталу

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Теоретичні основи розробки освітніх веб-порталів
4.2 Шаблони проектування mvc при розробці освітнього порталу та реалізація принципів solid
4.3 Реалізація системи
4.4 Розрахунок кількості управляючої інформації.

5. Перелік графічного матеріалу
1 34 рисунки
2. 1 таблиця
3. 4 додатки
4. 20 використаних джерел

6. Дата видачі завдання_____

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Визначення тематики, вибір наукового керівника, уточнення теми | 10.12.2020 | |
| 2 | Розробка та складання плану кваліфікаційної бакалаврської роботи | 12.01.2021 | |
| 3 | Підготовка 1 розділу | 17.02.2021 | |
| 4 | Підготовка 2 розділу | 09.04.2021 | |
| 5 | Підготовка 3 розділу | | |
| 6 | Висновки | | |
| 7 | Підготовка остаточного варіанту роботи | | |
| 8 | Написання відзиву науковим керівником | | |
| 9 | Оформлення та представлення роботи на кафедру та попередній захист | | |
| 10 | Зовнішня рецензія | | |
| 11 | Підготовка доповіді, презентації та ілюстративного матеріалу | | |
| 12 | Захист кваліфікаційної бакалаврської роботи | | |

Студент _____ _____
( підпис ) (прізвище та ініціали)

Керівник роботи _____
підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Текстова частина дипломної роботи містить 55 сторінок, 34 рисунка, 1 таблицю, 20 джерел, 3 додатки

ОСВІТНІЙ ПОРТАЛ, ДОДАТКОВУ ОСВІТУ, ВЕБ-РОЗРОБКА, ОСВІТНІЙ ТРЕК, МЕТОДИКА ПОБУДОВИ ОСВІТНІХ ТРЕКІВ, СПОСОБИ ПОБУДОВИ ОСВІТНІХ ТРАЄКТОРІЙ, МОДЕЛЬ ФОРМУВАННЯ КОМПЕТЕНЦІЙ, СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ, СИСТЕМНИЙ АНАЛІЗ, ПРОЕКТУВАННЯ ВЕБ-ПОРТАЛІВ, ПРИНИЦИПИ SOLID, МОДЕЛЬ MVC, ПРОЕКТУВАННЯ.

Об'єкт дослідження: освітній веб-портал

Мета дослідження - аналіз, проектування і реалізація освітнього порталу додаткової освіти школярів

У процесі дослідження проведено аналіз предметної області та теоретичних основ розробки освітніх веб-порталів. Виявлено характеристики і основні функції, якими повинен володіти сучасний освітній портал. Проведено аналіз конкурентних систем орієнтованих на додаткову освіту школярів. Складено портрети потенційних користувачів системи. Проведена вартісна і часова оцінка проекту. Складено календарний план проекту. Здійснено вибір методології розробки ПО. Розроблено методику побудови освітніх траєкторій. Проведено збір та аналіз вимог до системи. Здійснено вибір архітектури системи. Описано основні сценарії використання системи, поля сутностей бази даних. Складено технічне завдання. Обрані інструменти реалізації системи і технологічний стек.

В результаті розроблена перша версія освітнього порталу.

# ЗМІСТ

## ВСТУП

З кожним днем інформаційні технології все більше впроваджуються в наше життя. Нові винаходи в цій сфері створюють можливості для ефективної комунікації з будь-якої точки світу. Сьогодні багато звичних сфери людської діяльності зазнають змін - сфера освіти не є винятком. Навчання онлайн являє собою новий щабель і новий рівень освітнього процесу. Обсяг інформації, яка доступна в інтернеті і яку можна використовувати в освітніх цілях, по суті безмежний. І в цьому контексті саме оперативну взаємодію всіх учасників освітнього процесу в режимі віддаленого доступу грає ключову роль.

**Актуальність** розвитку системи додаткової освіти школярів в сучасних соціокультурних умовах обумовлена різким прискоренням процесів виробництва, споживання і накопичення інформації. Сучасна система основного шкільної освіти, заснована на державних стандартах, часто не встигає адаптуватися до змін, що відбуваються і не може в повному обсязі підготувати учнів до майбутньої професійної діяльності.

Як високо адаптивного інструменту, який дозволяє брати участь в процесі освіти необмеженому числу осіб, незалежно від їх місцезнаходження, на перший план виходять освітні веб-портали.

Наявні на даний момент портали або є типовими інформаційними ресурсами, де просто розміщені матеріали для роботи школярів, або спрямовані на допомогу по основній шкільній програмі. Ресурсів, які спрямовані на додаткову освіту школярів, що дозволяють вибудовувати індивідуальні траєкторії розвитку з прицілом на майбутню професійну діяльність і реальні проекти, на даний момент немає, хоча і робляться численні спроби їх створення.

**Мета дипломної роботи** - розробка освітнього порталу додаткової освіти школярів.

**Об'єкт дослідження:** освітній веб-портал.

В процесі дослідження вирішувалися наступні **завдання**:

— ознайомлення з сучасними інтернет-технологіями та їх використання в цій розробці (PHP(Laravel), JavaScript, HTML5, CSS3);

— вивчення програмного інструментарію, що застосовується для розробки та створення Web-сайтів;

— виявлення та облік методів і способів подання на Web сторінках різних видів інформації, що не перешкоджають їх доступність;

— ознайомлення з основними правилами та рекомендаціями щодо розробки та створення Web-сайтів і неухильне слідування їм на практиці;

— визначення структури Web-сторінок;

— розробка карти web-сайту;

— вибір стратегії розробки і створення освітнього-пізнавального Web-сайту.

# 1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ОСВІТНІХ ВЕБ-ПОРТАЛІВ

## 1.1 Поняття, сутність, види та завдання освітніх веб-порталів

Для розуміння того що представляє собою сучасний освітній веб-портал необхідно визначити сутність, яка лежить в його основі. Поняття «сайт», хоча і є англіцизмом, зараз стало дуже поширеним і увійшло в нашу лексику, що пов'язано з розвитком мережі Інтернет і з її активним використанням в усіх сферах життєдіяльності людини, в тому числі, освіті. Ігнатова Н.Г. визначає сайт (від англ. Site - місце, місце розташування, позиція) як сукупність сторінок, об'єднаних однією спільною темою, дизайном, мають взаємопов'язану систему посилань, розташованих в мережі Інтернет [3, с. 28]. У статті Н.В. Бужинської сайт визначено як сукупність пов'язаних гіперпосиланнями сторінок, об'єднану однією спільною темою, дизайном, системою навігації, розташовану в Інтернеті під певним адресою [2, с.18].

Синонімами «сайту» стали поняття «інтернет-сайт», «веб-сайт», "Інтернет ресурс". Інтернет-сайту відповідає одне, так зване, доменне ім'я, за яким його можна знайти в інтернеті. Саме це ім'я бере участь в так званій «засланні» на сайт. У зв'язку з цим, А.Г. Бабаєв дає наступне визначення сайту - це «структурована інформаційна одиниця всесвітньої павутини, яка може містити як одну, так і величезна кількість сторінок». Наприклад, сайт компанії IBM містять кілька тисяч сторінок, а сайт Томського політехнічного університету - кілька сотень.

За доступності А.Г. Бабаєв виділяє відкриті для всіх користувачів, напіввідкриті і закриті сайти. Частина інформації другої групи сайтів відкрита для всіх, а частина прихована. Для її перегляду на сайті необхідна безкоштовна або

платна реєстрація [1, с. 27]. За величиною і рівнем розв'язуваних завдань сайти діляться на такі групи, як:

— інформаційні сайти-візитки, що містять трохи інформації і складаються з декількох сторінок;

— тематичні, вузькоспрямовані сайти, що представляють користувачеві інформацію по якійсь одній темі;

— багатофункціональні сайти (портали), що містять, крім інформації, засоби для спілкування користувачів, чати, форуми і т.д.

Освітній сайт на сьогоднішній день найпоширеніший з видів освітніх Інтернет-ресурсів і найпопулярніший серед представників педагогічних працівників для реалізації.

Хуторський А.В. визначає освітній сайт як «цілісну, концептуально обгрунтовану і структурно вибудувану систему, що об'єднує в собі взаємопов'язані між собою веб-сторінки, зміст яких підпорядковане загальній ідеї і виражено в конкретних цілях і завданнях кожної з них» [6, с. 53].

Основні завдання освітнього сайту, які також можна віднести і до завдань інших видів освітніх Інтернет-ресурсів, Н.Г. Ігнатова бачить в наступному:

забезпечення відкритості освітнього процесу і висвітлення його в мережі Інтернет;

створення умов для взаємодії всіх учасників освітнього процесу (педагога, учнів і їх батьків);

— оперативне і об'єктивне інформування учасників освітнього процесу про результати навчання;

— підвищення авторитету вчителя за допомогою уявлення досягнень учнів;

— стимулювання творчої активності педагога та учнів;

— активне використання педагогом ІКТ для вирішення завдань модернізації освіти;

— підвищення ролі інформатизації освіти, сприяння створенню в регіоні єдиної інформаційної інфраструктури.

Інший вид освітніх ресурсів - бази знань (від англ. Knowledge base). Вони являють собою сховища великого обсягу освітніх файлів і документів. У базі знань можуть міститися спеціалізовані статті, довідники, енциклопедії. Зазвичай в база даних є вбудовані засоби пошуку інформації.

Системою дистанційної освіти називається Інтернет-ресурс, в якому навчальний процес організований від стадії складання навчального плану і закінчуючи отриманням засвідчує документа, наприклад, диплом або сертифікат. Такі системи зазвичай містять електронні підручники, програми навчання, віртуальні семінари, системи контролю успішності.

Найскладніший і осяжний вид освітніх Інтернет-ресурсів - це освітній портал. Освітній портал - це найбільш перспективний напрямок застосування інформаційно-комп'ютерних технологій в освіті. Існує ряд визначень порталу, часто в науковій і технічній літературі під порталом розуміється:

— Веб-сайти, орієнтовані на певні аудиторії і спільноти, які забезпечують: об'єднання інформаційного наповнення та доставку важливої для даної аудиторії інформації; спільну роботу і колективні е послуги; доступ до послуг і додатків для обраної аудиторії, що надається на основі суворої персоналізації [2];

— комп'ютерна система (додаток, мультисервісний сервер), що забезпечує персоніфікований і зручний інтерфейс, можливість людям знаходити і взаємодіяти з іншими людьми, знаходити і використовувати інформацію відповідно до своїх інтересів [4];

— інформаційний вузол, сукупність тематичних сайтів, об'єднаних пошуковою системою, основна функція якого полягає в забезпеченні підключення клієнтів - відвідувачів порталу, до відповідних джерел інформації та т. П.

У загальному вигляді портали зазвичай позиціонують як відправні точки для користувачів, орієнтованих на певну тематичну область. Можна помітити істотні відмінності у визначеннях порталу, але не дивлячись на це, можна виділити загальні моменти, які відображають сутність даного поняття. Портал - це єдина інтегрована точка ефективного всебічного необмеженого доступу до інформації, додатків і людям. Таким чином, портальна технологія дозволяє максимально наблизити ресурси до користувачів, забезпечує інтеграцію інформаційної сутності організації, організовує відносини всередині робочих та інформаційних груп, створюючи умови для єдиного інформаційного простору [3].

## 1.2 Характеристики і функції освітнього порталу

Сучасні портали є складними і великими інформаційними системами, розробка яких вимагає глибокого і різнобічного концептуального дослідження.

Важливими характеристиками порталу, в тому числі освітнього є:

— персоналізація для кінцевих користувачів - портал повинен дозволяти налаштовувати свій зовнішній вигляд і / або вміст додатків для кожного користувача індивідуально;

— доступ користувача до інформаційних ресурсів повинен бути організований в найбільш зручному, консолідованому вигляді;

— портали повинні забезпечувати ідентифікацію користувача, т. Е. Підтримувати аутентифікацію, єдину реєстрацію на сервері, створення карти прав доступу і т. д.;

— відстеження виконання робіт - ця характеристика особливо важлива для персоніфікації порталу, яка встановлюється на початку його використання користувачем і наростає в міру накопичення інформації про його інтереси і схильності;

— активний доступ і відображення інформації зі сховища даних;

— локалізація і виявлення потрібних людей і інформації - використовувані пошукові механізми повинні забезпечувати як пасивне інформування та виявлення, так і кошти активного виявлення експертів, спільнот і контенту, пов'язаного з певною тематикою.

Отже, ми виділили такі вимоги до порталу при його розгортанні: обслуговування великого числа користувачів (студентів вузів); широкий спектр інформації; підтримка основних мережевих форматів; широкі можливості персоналізації; реалізація зручних і ефективних пошукових механізмів, оцінка достовірності і повноти отриманих даних; забезпечення захисту інформації, що зберігається з використанням програмних і фізичних способів забезпечення безпеки; інтеграція - забезпечення можливості взаємодії користувачів з усіма додатками і інформаційними ресурсами через єдиний інтерфейс; розбивка зберігається на категорії - категоризація, автоматизовані процедури категоризації результатів пошуку; додатки інтелектуального аналізу - системи управління знаннями [6].

Однією з основних форм, що визначають уявлення і розповсюдження контенту на порталі, є персоналізація. Для цього використовуються механізми фільтрації інформації та аналізу роботи користувача, з їх допомогою вдається визначити ту область, яка може зацікавити його. Відвідувачам порталу направляються особисті вітання, рекламні оголошення, надається можливість налаштування інтерфейсу порталу, регулярної доставки певної інформації і т. Д.

В основному в даний час використовується два методи персоналізації:

— з використанням правил - на основі введеної інформації до реєстраційної картки особистої інформації про себе та свої інтереси, розробляється набір правил, які виконуються в процесі обробки різних запитів користувача на надання доступу до певних інформаційних ресурсів, до елементів інтерфейсу і т. Д .;

— на основі фільтрів використовуються складні алгоритми категоризації і надання контенту на основі аналізу поведінки користувача (до якої інформації він звертається, які сайти відвідує і т. д.).

Таким чином, незважаючи на відмінності в організації, обидві системи персоніфікації об'єднує механізм розмежування рівнів доступу до баз даних порталу.

Сервіси, які підтримуються всіма порталами і являють собою загальну платформу, є основою для більш ефективного управління додатками та інформацією, що надходить з різних джерел, зменшують завантаження персоналу і адміністративні витрати. Серед основних сервісів портальної технології можна виділити: сервіси спільнот, які організовують портальні спільноти і забезпечують доступ до сервісів через реєстрацію користувачів і політики безпеки.

Можливість інтеграції сервісів і архітектури є одним з основних переваг відкритою портальної технології, так як дозволяє організаціям використовувати вже наявні веб-сумісні програми. Але в строгому значенні понять слід розрізняти освітній портал (портал навчання) і інформаційний портал системи освіти. Перший на додаток до функцій організації доступу до освітньої інформації має також функціями створення та контролю знань, а також підтвердження досягнутого освітнього рівня, т. Е. Реалізує функції навчання. Якщо ж в порталі такої функції навчання немає, то такий портал є лише інформаційним порталом.

З огляду на, що освітня діяльність базується на використанні великих обсягів інформації, відповідно саме система порталів, як засіб уявлення, поширення, систематизації, структуризації та уніфікації інформаційних ресурсів Інтернет, може забезпечити найбільш раціональний спосіб використання освітніх ресурсів.

Таким чином, освітній портал є багатофункціональним сайтом зі своєю специфікою. Сайт повинен володіти рядом специфічних функцій, а його наповнення та структура визначається цілями створення і цільовою аудиторією. В умовах єдиного інформаційного простору сайт стає невід'ємною частиною

освітнього процесу, який використовується як засіб дистанційної підтримки освіти і консультування учнів.

### 1.3 Основні етапи створення освітнього порталу

Освітній портал - це інформаційна система, отже, до нього може бути застосована методологія проектування інформаційних систем. В основі методології проектування інформаційних систем лежить поняття життєвого циклу (ЖЦ) системи, яке можна уявити, як набір етапів і процесів, які виконуються на цих етапах.

При формальному описі життєвого циклу системи для кожного окремого етапу визначаються:

—склад і послідовність виконуваних робіт;

—отримані результати;

—методи і засоби, необхідні для виконання робіт;

—ролі та відповідальність учасників.

Даний підхід дозволяє спланувати та організувати процес колективної розробки і забезпечити управління цим процесом.

Зазвичай виділяють наступні етапи створення ІС:

—формування вимог до системи;

—проектування;

—реалізація;

—тестування;

—введення в дію;

—експлуатація і супровід.

Коротко розглянемо перші чотири етапи створення системи.

Формування та аналіз вимог. Даний етап є першим і найважливішим, оскільки саме на цьому етапі фактично дається відповідь на питання: «Що

майбутня система повинна робити?». Тобто формуються вимоги до системи, які повинні точно і повно відображати мету і завдання замовника, що визначає всі подальші етапи розробки.

Список вимог до розроблюваної системі повинен включати [11]:

—сукупність умов, при яких передбачається експлуатувати майбутню систему (апаратні і програмні ресурси, що надаються системою;

—зовнішні умови її функціонування;

—склад людей і робіт, які мають до неї відношення;

—опис виконуваних системою функцій;

—обмеження в процесі розробки (директивні терміни завершення окремих етапів, наявні ресурси, організаційні процедури і заходи, що забезпечують захист інформації).

На цьому етапі визначається:

—архітектура системи, її функції, зовнішні умови, розподіл функцій між апаратурою і ПО;

—інтерфейси і розподіл функцій між людиною і системою;

—вимоги до програмних і інформаційних компонентів ПЗ, необхідні апаратні ресурси, вимоги до БД, фізичні характеристики.

Проектування. Даний етап є другим і фактично дає відповідь на питання: "Яким чином буде працювати система, щоб задовольняти пред'явленим вимогам?».

На даному етапі досліджується структура системи та логічні її взаємозв'язку. В результаті виходить логічна модель системи, опис моделі даних і опис модулів системи.

Реалізація. На етапі реалізації здійснюється створення програмного забезпечення системи, установка технічних засобів, розробка експлуатаційної документації.

Тестування Етап тестування зазвичай виявляється розподіленим в часі і проводиться в міру готовності модулів системи. При завершенні робіт зі створення системи проводиться фінальний тест, який моделює реальні бізнес-процеси, щоб показати замовнику відповідність системи заявленим вимогам.

Таким чином, ми розглянули основні етапи розробки освітнього порталу. Існують різні методології розробки ПО, засновані на концепції життєвого циклу. Від вибору конкретної методології безпосередньо залежить успішність проекту.

## 1.4 Засоби розробки освітнього порталу

Для створення і супроводу динамічних сайтів було використано фреймворк - систему управління вмістом. В даний час популярними фреймворками є Laravel, Symfony, Codeigniter, Yii2. На основі цих фреймворків можна створювати функціональні і легко керовані PHP-сайти.

Для створення сайту використані знання з HTML, CSS, PHP та навички при роботі з Laravel і сервером Apache.

### 1.4.1 Мова розмітки гіпертекстових документів HTML5

HTML (англ. HyperText Markup Language — Мова розмітки гіпертекстових документів) — основана на SGML текстова мова розмітки, призначена для маркування документів, що містять текст, зображення, гіперпосилання, тощо. HTML-документи лежать в основі Веб, і відображаються із допомогою веб-браузерів. Разом із видимою інформацією, HTML-документи містять додаткові метадані, такі як, наприклад, мова тексту, автор документа, стислий підсумок. Мова розмітки розроблялась консорціумом W3C, остання версія — 4.01,

очікується, що HTML буде замінена розширюваною мовою розмітки гіпертексту (XHTML) [12].

Найпростіша веб-сторінка складається з текстових блоків, декількох рисунків, горизонтальних розмежувальних ліній та гіперпосилань. Більш складні веб-сторінки містять фрейми, елементи керування, динамічні ефекти та анімовані об'єкти.

Крім свого стандартного застосування таблиці в html-мові також використовують, якщо потрібно розташувати якийсь текст чи об'єкт у певному місті на веб-сторінці [13].

### 1.4.2 Каскадні таблиці стилів CSS3

Каскадні таблиці стилів (CSS) є потужним інструментом для керування зовнішній вигляд вашого веб-сайту, вплине на уявлення документа або набору документів.

Переваги CSS:

— CSS дозволяє значно багатший, ніж документ виступу HTML все дозволено, навіть у розпал презентаційних запал HTML;

— CSS дозволяє задати кольору на текст і на тлі будь-якого елементу;

— CSS дозволяє створювати рамки навколо будь-якого елемента, а також збільшення або зменшення простору навколо них;

— CSS дозволяє змінити спосіб текст пишеться з великої літери, прикрашені (наприклад, основний), розташованих на відстані, і навіть будь воно відображається на всіх.

І CSS дозволяє виконувати багато інших ефектів. За допомогою CSS, ви також можете створювати спеціальні ефекти, включаючи текст перекидання, а також контролювати розміщення графіки та дизайну макета сторінки.

Каскадні таблиці стилів являє собою набір інструкцій, який вказує веб-браузер, як представити або дисплей, різні HTML-елементи, такі як, який шрифт використовувати, який розмір тексту абзацу повинен бути, який колір тексту заголовка має бути, або Не повинно бути колір фону стосовно до конкретних елементів, і так далі. Ви можете використовувати кілька таблиць стилів, щоб налаштувати дисплей вашого веб-сайту в різних браузерах, різних платформ і різних пристроїв [16].

### 1.4.3 Мова програмування PHP, Javascript

PHP (англ. PHP: Hypertext Preprocessor — «Інструмент для створення персональних веб-сторінок») — мова програмування, створена для генерування HTML-сторінок на веб-сервері і роботах з базами даних.

В даний час, підтримується переважною більшістю хостинг- провайдерів. В області програмування для Мережі, PHP — одна з популярних скріптових мов (поряд з JSP, Perl і ASP) завдяки своїй простоті, швидкості виконання, багатій функціональності і поширенню початкових кодів [14].

PHP відрізняється наявністю ядра і модулів, що підключаються для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і тому подібне. Існують сотні розширень проте в стандартне постачання входить лише декілька десятків тих, що добре зарекомендували себе. Інтерпретатор PHP підключається до веб-серверу або через

модуль, створений спеціально для цього сервера (наприклад, для Apache або IIS), або як Cgi-розширення.

### 1.4.4 Фреймворк Laravel

Laravel - безкоштовний веб-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC (англ. Model View Controller - модель-уявлення-контролер). Laravel випущений під ліцензією MIT.

Архітектуру додатків Laravel визначають як full-stack фреймворк, оскільки він може працювати з усіма можливими частинами веб-додатки. До таких частин можна віднести: веб-сервіси, управління базами даних, генерація HMTL, маршрутизація, кешування, аутентифікація, локалізація, управління групами користувачів, збірка css і js файлів, інкапсуляція ресурсів і багато іншого. Таким чином вертикально інтегроване середовище веб-розробки забезпечує кращий досвід для розробника.

З Laravel розробник може взаємодіяти як через вбудовану утиліту командного рядка, яка управляє середовищем проекту Laravel, так і просто за допомогою настройки конфігураційних файлів вручну. Як правило обидва ці способи використовуються разом. Одна з цікавих особливостей Laravel полягає в тому, що вона накладає деякі досить серйозні обмеження на структуру веб-додатків. Але, як не дивно, ці обмеження спрощують процес розробки. Laravel відрізняється від інших вертикально інтегрованих середовищ тим, що вона конфігурацій воліє конвенції.

У той час як в деяких веб-фреймворк, написаних на Java, Python або PHP, часто потрібно редагування дуже великого обсягу файлів з конфігураціями, то для початку роботи з Laravel ніяких налаштувань практично не потрібно, хіба що крім декількох рядків на PHP. Варто зазначити, що велика кількість конфігураційних файлів в Laravel написано на PHP. Такий підхід в уникненні файлів конфігурації

дозволяє додаткам, написаним під різні потреби, підтримувати впізнавану структуру коду.

З огляду на все вище сказане, не дивно, що проекти під Laravel мають практично однакову файлову структуру - таку, в якій кожен функціональний блок має своє визначене місце. Така структура зовсім не обов'язкова для повноцінної роботи програми, але ненав'язливе пропозицію такого способу зберігання файлів в проекті, гарантує більш-менш уніфіковану архітектуру серед всіх додатків Laravel, що має спростити досвід роботи з цим фреймворком в цілому. Якщо все відкриті проекти, навчальні курси і підручники будуть засновані на однаковій файлової структурі, це значно знизить поріг входження для нових програмістів. Нижче представлена файлова структура розробляється.

*Рисунок 1.4.1 Файлова структура*

Як видно, стандартна файлова структура складається з великої кількості каталогів. Таке багатство піддиректорій може спочатку збентежити, але ми розберемо їх одну за одною.

Нижче наведено короткий виклад функцій кожного з цих файлів і папок:

/ app / Складається з контролерів MVC моделі, моделей ORM і утиліт консолі.

/ Bootstrap / Містить в собі файли, які необхідні для запуску і роботи ядра Laravel.

/ Config / Тільки за назвою можна визначити, що в цій папці зберігаються файли для настройки всіляких модулів Laravel:

починаючи з бази даних і кешування, закінчуючи модулем відправки листів і файлової системою.

/ Database / В цій папці зберігаються файли міграцій, сідування і генерації баз даних з моделей.

/ Node_modules / Папка, в якій зберігаються підключаються JS модулі, встановлені за допомогою утиліти nvm.

/ Public / Тільки ця папка на сервері доступна зовнішнього світу. На цей каталог повинен посилатися веб-сервер. Він містить завантажувальний файл index.php, який запускає ядро Laravel.

Цей каталог також може бути використаний для зберігання будь-яких загальнодоступних статичних файлів, таких як CSS, JavaScript, зображення та інші.

/ Python / Папка, створена для зберігання в ній python-утиліт, призначених для роботи з API Kudago. / Resources / В цій папці знаходяться ресурси для роботи з фронтенда: вихідні JavaScript і css файли для збірок, файли локалізації та файли представлень.

/ Routes / Тут зберігаються файли для конфігурації модуля маршрутизації, а саме web.php, робота якого буде детально описана.

/ Storage / Ця папка призначена для зберігання генерованих роботою Laravel файлів: сесій, кеша і балок.

/ Tests / Як випливає з назви, в цій папці зберігаються файли модульних тестів.

/ Vendor / Це місце для всіх сторонніх частин програми. У типовому додатку Laravel в нього включається вихідний код Laravel, його залежності, а також плагіни, які містять додаткові попередньо упаковані функції. З цього файлу Laravel бере environment-змінні, наприклад, тут, за допомогою зміни значення змінної DEBUG, можна задати стан Laravel додатки - production або debug. gulpfile.js Файл настройки модуля автоматизації завдань збірки Gulp.

### 1.4.5 Веб сервер Apache

Найпоширеніший веб-сервер в світі - це Apache. За даними компанії Netcraft, загальне число веб-узлів, що працюють під його управлінням, до кінця 1998 р. досягло 2 млн. (55% загального числа вузлів) і постійно росте. Для порівняння: на долю серверів Microsoft доводиться 25%, Netscape - 7%. Будучи безкоштовною відкритою програмою, призначеною для безкоштовних же Unix-систем (FREEBSD, Linux і ін.), Apache по функціональних можливостях і надійності не поступається комерційним серверам, а широкі можливості конфігурації дозволяють побудувати його для роботи практично з будь-якою конкретною системою. Існують локалізації сервера для різних мов, у тому числі і для російської.

Історично склалося так, що російські тексти в Internet можуть бути представлені в різних кодуваннях, з яких найбільш поширені koi8-r (або просто koi8) і Windows-1251: з першою працюють більшість серверів і робочих станцій під управлінням Unix, друга є стандартною для всіх версій Windows. Оскільки кодування Windows-1251, природно, застосовується на переважній більшості клієнтських машин, частка тих, хто подорожує по російській частині WWW, використовуючи koi8, не перевищує зараз 5%.

Проте в цьому кодуванні зберігаються документи на багатьох Unix- серверах, в ній найчастіше передаються поштові повідомлення і практично завжди - листи в телеконференції, з нею ж працюють багато російськомовних каналів IRC (до речі, абревіатура KOI розшифровується як ("код обміну інформацією"). Щоб вирішити проблеми, що виникають при неспівпаданні кодувань тексту на сервері і клієнтській машині, і був створений російський модуль APACHE-RUS для веб-сервера Apache [15].

# 2 ШАБЛОНИ ПРОЕКТУВАННЯ MVC ПРИ РОЗРОБЦІ ОСВІТНЬОГО ПОРТАЛУ ТА РЕАЛІЗАЦІЯ ПРИНЦИПІВ SOLID

## 2.1 Model-View-Controller

У фундаментальному патерні MVC (Model-View-Controller) є три основні складові: View (представлення, відображення, користувацький інтерфейс), Model (модель, бізнес логіка) і Controller (контролер, вміщує в собі логіку за зміну моделі при взаємодії користувача з представленням, реалізує Use Case). Стандартну схему архітектури «Модель-Представлення-Контролер» подано на Рисунку 2.1.


*Рисунок 2.1 - Схема архітектури MVC*

MVC знайшов застосування в розробці багатьох програм, дав розвиток новим технологіям і кожного дня полегшує життя програмістам. Основна ідея цього шаблону полягає в тому, що і представлення, і контролер залежать від моделі, але модель від цих двох компонентів зовсім не залежить.

Ця особливість і дозволяє розробляти, змінювати і тестувати бізнес логіку, нічого не знаючи про представлення та контролери. В свою чергу, контролер не повинен нічого знати про відображення і для одно- го представлення можна перемикати контролери. Так само один і той же контролер може використовуватися для різного інтерфейсу (наприклад, контролер, який буде виконуватися, може залежати від статусу користувача, який увійшов в систему). Користувач бачить відображення, взаємодіє з ним, ці дії частина представлення

пере- направляє контролеру і підписується на зміни даних моделі. Контролер в свою чергу виконує певні функції з моделлю даних, відображення отримує останній стан моделі і оновлює представлення користувачеві.

Головною метою архітектурного шаблону MVC є створення гнучкого дизайну програмного забезпечення, який в подальшому полегшує розширення та зміни про- грами, а також надає можливість повторно використовувати окремі компоненти. Окрім того використання даного патерну у великих системах сприятиме впорядку- ванню їх структури та робитиме ці системи зрозумілішими за рахунок зменшення складності [6]. Розглянемо більш детально кожен елемент архітектурного патерну MVC.

### 2.1.1 Модель

Модель - центральний компонент MVC. Відповідальний за поведінку додатку, яка не залежить від користувацького інтерфейсу. Модель вміщує в собі функції керу- вання, збереження даних та їх структури, керування логікою додатку, відповідає за алгоритми, розрахунки, внутрішні устрої системи.

Модель - містить бізнес-логіку додатка і включає методи вибірки (це можуть бути методи ORM), обробки (наприклад, правила валідації) і надання конкретних даних, що часто робить її дуже товстою, що цілком нормально.

Модель не повинна безпосередньо взаємодіяти з користувачем. Всі змінні, що відносяться до запиту користувача повинні оброблятися в контролері.

Модель не повинна генерувати HTML або інший код відображення, який може змінюватися в залежності від потреб користувача. Такий код повинен оброблятися в видах.

Одна і та ж модель, наприклад: модель аутентифікації користувачів може використовуватися як в призначеній для користувача, так і в адміністративній частині програми. В такому випадку можна винести загальний код в окремий клас

і успадковуватися від нього, визначаючи в спадкоємців специфічні для подпріложеній методи.

### 2.1.2 Представлення

Представлення - модуль, відповідальний за виведення інформації, яка надходить із системи або в систему, відображення даних моделі користувачу, внаслідок зміни моделі. Для однієї і тієї ж інформації можуть співіснувати одночасно декілька представлень (виглядів). Представлення не відповідає за обробку введених даних користувача.

Представлення - використовується для завдання зовнішнього відображення даних, отриманих з контролера і моделі.

Види мають HTML-розмітку і невеликі вставки PHP-коду для обходу, форматування і відображення даних. Чи не повинні безпосередньо звертатися до бази даних. Цим повинні займатися моделі. Чи не повинні працювати з даними, отриманими із запиту користувача. Це завдання має виконувати контролер.

Може безпосередньо звертатися до властивостей і методів контролера або моделей, для отримання готових до висновку даних.

Види зазвичай поділяють на загальний шаблон, що містить розмітку, загальну для всіх сторінок (наприклад, шапку і підвал) і частини шаблону, які використовують для відображення даних, що виводяться з моделі або відображення форм введення даних.

### 2.1.3 Контролер

Контролер - модуль керування введеням і виведенням даних, забезпечує “зв’я- зок” між користувачем та системою. Контролер передає дані від користувача до си- стеми і навпаки. Отримуючи вхідні дані, він перекладає їх на команди для інших ча- стин системи. Для реалізації необхідної функції

використовує дані з компонентів вигляду та моделі [7]. Контролер відповідальний за передану інформацію в систему, адже на основі введених даних саме він повинен визначити:

— виконати запит на повторне введення даних та виводити повідомлення про помилку (модуль представлення має оновити вигляд сторінки, виведену інформацію);

— передати введені дані в модель системи.

Контролер - сполучна ланка, що з'єднує моделі, види і інші компоненти в робоче додаток. Контролер відповідає за обробку запитів користувача. Контролер не повинен містити SQL-запитів. Їх краще тримати в моделях. Контролер не повинен містити HTML і інший розмітки. Її варто виносити в види.

У добре спроектованому MVC-додатку контролери зазвичай дуже тонкі і містять тільки кілька десятків рядків коду. Чого, не скажеш про Stupid Fat Controllers (SFC) в CMS Joomla. Логіка контролера досить типова і велика її частина виноситься в базові класи.

Моделі, навпаки, дуже товсті і містять велику частину коду, пов'язану з обробкою даних, тому що структура даних і бізнес-логіка, що міститься в них, зазвичай досить специфічна для конкретного додатка.

### 2.1.4 Недоліки патерну MVC

Необхідно використовувати більшу кількість ресурсів. Складність обумовлена тим, що всі три фундаментальні компоненти абсолютно незалежні між собою та взаємодіють виключно шляхом передачі даних. Контролер завжди повинен заван- тажувати (і за необхідності створювати) всі можливі комбінації змінних і передавати їх в модель. Компонент моделі, в свою чергу, має завантажити всі дані для візуаліза- ції і передати їх у представлення. Наприклад, в модульному методі, модуль має мож- ливість безпосередньо візуалізувати дані і обробляти змінні без завантаження їх в окремі відділи пам'яті.

Ускладнений алгоритм поділу програми на модулі. У концепції MVC наявність трьох блоків (Модель, Представлення, Контролер) задано жорстко. Відповідно кожен функціональний компонент має складатися з трьох секцій, що в свою чергу, дещо ускладнює та навантажує архітектуру функціональних модулів програми.

Погіршений механізм розширення функціоналу. Проблема досить схожа на вищезазначену. Недостатньо просто написати функціональний модуль і застосувати його в одному місці програми. Кожна функціональний частина повинна складатися з трьох модулів, і кожен з цих модулів повинен бути підключеним у відповідному блоці.

### 2.1.5 Переваги патерну MVC

Єдина концепція системи. Безсумнівною перевагою MVC є єдина глобальна архітектура програми. Навіть в складних системах, розробники (як ті, які від самого початку працювали над розробкою системи, так і ті, які нещодавно приєдналися) можуть легко орієнтуватися в програмних блоках. Так, наприклад, якщо виникла помилка в логіці обробки даних (компонент моделі), розробник одразу відкидає оби- два інші компоненти програми (контролер і представлення) і займається поглибле- ним вивченням третього компоненту (моделі). Дана властивість значно спрощує локалізація проблеми і витрати часу на пошук необхідного для виправлення компоненту зменшуються.

## 2.2 Реалізація принципів SOLID

### 2.2.1 Принцип S: Single Responsibility Principle.

Принцип єдиної відповідальності

Клас повинен бути відповідальний лише за щось одне. Якщо клас відповідає за вирішення декількох завдань, його підсистеми, що реалізують рішення цих задач, виявляються пов'язаними один з одним. Зміни в одній такій підсистемі ведуть до змін в іншій.

Наприклад, розглянемо цей код на Рисунку 2.2.1:

```
class Animal {
    constructor(name: string){ }
    getAnimalName() { }
    saveAnimal(a: Animal) { }
}
```

Рисунок 2.2.1 – приклад коду

Клас Animal, представлений тут, описує якусь тварину. Цей клас порушує принцип єдиної відповідальності.

Як саме порушується цей принцип?

Відповідно до принципу єдиної відповідальності клас повинен вирішувати лише якусь одну задачу. Він же вирішує дві, займаючись роботою зі сховищем даних в методі saveAnimali маніпулюючи властивостями об'єкта в конструкторі і в методі getAnimalName.

Як така структура класу може привести до проблем?

Якщо зміниться порядок роботи зі сховищем даних, що використовуються додатком, то доведеться вносити зміни в усі класи, які працюють зі сховищем.

Така архітектура не відрізняється гнучкістю, зміни одних підсистем зачіпають інші, що нагадує ефект доміно.

Для того щоб привести вищенаведений код у відповідність до принципу єдиної відповідальності, створимо ще один клас, єдиним завданням якого є робота зі сховищем, зокрема - збереження в ньому об'єктів класу Animal на Рисунку 2.2.2:

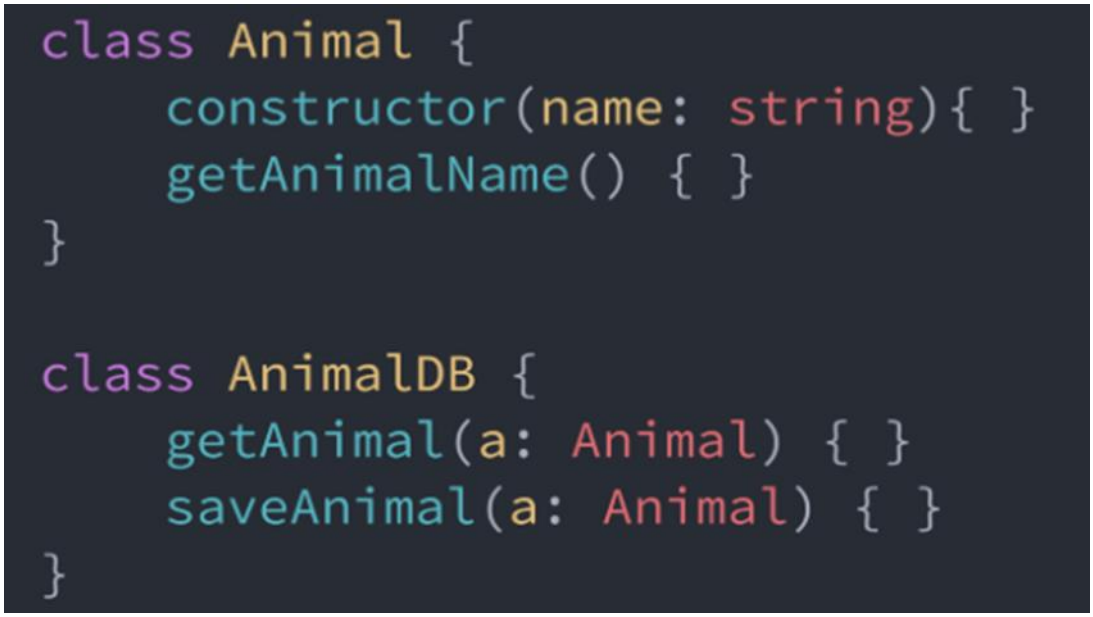



Рисунок 2.2.1 – приклад коду

Ось що з цього приводу говорить Стів Фентон: «Проектуючи класи, ми повинні прагнути до того, щоб об'єднувати споріднені компоненти, тобто такі, зміни в яких відбуваються за одними і тими ж причинами. Нам слід намагатися розділяти компоненти, зміни в яких викликають різні причини ».

Правильне застосування принципу єдиної відповідальності призводить до високого ступеня зв'язності елементів всередині модуля, тобто до того, що завдання, які вирішуються всередині нього, добре відповідають його головної мети.

### 2.2.2 Принцип O: Open-Closed Principle

Принцип відкритості-закритості. Продовжимо роботу над класом Animal.

```
class Animal {
    constructor(name: string){ }
    getAnimalName() { }
}
```

*Рисунок 2.2.2 – приклад коду*

Ми хочемо перебрати список тварин, кожне з яких представлено об'єктом класу Animal, і дізнатися про те, які звуки вони видають. Уявімо, що ми вирішуємо це завдання за допомогою функції AnimalSounds:

Ми хочемо перебрати список тварин, кожне з яких представлено об'єктом класу Animal, і дізнатися про те, які звуки вони видають. Уявімо, що ми вирішуємо це завдання за допомогою функції AnimalSounds:

```
//...
const animals: Array<Animal> = [
    new Animal('lion'),
    new Animal('mouse')
];

function AnimalSound(a: Array<Animal>) {
    for(int i = 0; i <= a.length; i++) {
        if(a[i].name == 'lion')
            return 'roar';
        if(a[i].name == 'mouse')
            return 'squeak';
    }
}

AnimalSound(animals);
```

*Рисунок 2.2.2 – приклад коду*

Найголовніша проблема такої архітектури полягає в тому, що функція визначає те, який звук видає ту чи іншу тварину, аналізуючи конкретні об'єкти. Функція AnimalSoundне відповідає принципу відкритості-закритості, так

як, наприклад, при появі нових видів тварин, нам, для того, щоб з її допомогою можна було б дізнаватися звуки, що видаються ними, доведеться її змінити.

Додамо в масив новий елемент:

```
//...
const animals: Array<Animal> = [
    new Animal('lion'),
    new Animal('mouse'),
    new Animal('snake')
]
//...
```

*Рисунок 2.2.2 – приклад коду*

Після цього нам доведеться поміняти код функції AnimalSound:

```
//...
function AnimalSound(a: Array<Animal>) {
    for(int i = 0; i <= a.length; i++) {
        if(a[i].name == 'lion')
            return 'roar';
        if(a[i].name == 'mouse')
            return 'squeak';
        if(a[i].name == 'snake')
            return 'hiss';
    }
}

AnimalSound(animals);
```

*Рисунок 2.2.2 – приклад коду*

Як бачите, при додаванні в масив нового тваринного доведеться доповнювати код функції. Приклад це дуже простий, але якщо подібна архітектура використовується в реальному проекті, функцію доведеться постійно розширювати, додаючи в неї нові вирази if.

Можна помітити, що у класу Animaltепер є віртуальний метод makeSound. При такому підході потрібно, щоб класи, призначені для опису конкретних тварин, розширювали б клас Animali реалізовували б цей метод.

В результаті у кожного класу, що описує тваринного, буде власний метод makeSound, а при переборі масиву з тваринами в функції AnimalSoundдосить буде викликати цей метод для кожного елемента масиву.

Якщо тепер додати в масив об'єкт, що описує нова тварина, функцію AnimalSoundзмінювати не треба. Ми привели її у відповідність до принципу відкритості-закритості.

Розглянемо ще один приклад.

Уявімо, що у нас є магазин. Ми даємо клієнтам знижку в 20%, використовуючи такий клас:

```
class Discount {
    giveDiscount() {
        return this.price * 0.2
    }
}
```

*Рисунок 2.2.2 – приклад коду*

Тепер вирішено розділити клієнтів на дві групи. Улюбленим ( fav) клієнтам дається знижка в 20%, а VIP-клієнтам ( vip) - подвоєна знижка, тобто - 40%. Для того, щоб реалізувати цю логіку, було вирішено модифікувати клас наступним чином:

```
class Discount {
    giveDiscount() {
        if(this.customer == 'fav') {
            return this.price * 0.2;
        }
        if(this.customer == 'vip') {
            return this.price * 0.4;
        }
    }
}
```

*Рисунок 2.2.2 – приклад коду*

Такий підхід порушує принцип відкритості-закритості. Як видно, тут, якщо нам треба дати якоїсь групі клієнтів особливу знижку, доводиться додавати в клас новий код.

Для того щоб переробити цей код відповідно до принципу відкритості-закритості, додамо в проект новий клас, який розширює клас Discount. У цьому новому класі ми і реалізуємо новий механізм:

```
class VIPDiscount: Discount {
    getDiscount() {
        return super.getDiscount() * 2;
    }
}
```

*Рисунок 2.2.2 – приклад коду*

Якщо вирішено дати знижку в 80% «супер-VIP» клієнтам, виглядати це має так:

```
class SuperVIPDiscount: VIPDiscount {
    getDiscount() {
        return super.getDiscount() * 2;
    }
}
```

*Рисунок 2.2.2 – приклад коду*

Як бачите, тут використовується розширення можливостей класів, а не їх модифікація.

### 2.2.3 Принцип L: Liskov Substitution Principle

Принцип підстановки лісков

Мета цього принципу полягають в тому, щоб класи-спадкоємці могли б використовуватися замість батьківських класів, від яких вони утворені, не порушуючи роботу програми. Якщо виявляється, що в коді перевіряється тип класу, значить принцип підстановки порушується.

Розглянемо застосування цього принципу, повернувшись до прикладу з класом Animal. Напишемо функцію, призначену для повернення інформації про кількість кінцівок тварини.

```
//...
function AnimalLegCount(a: Array<Animal>) {
    for(int i = 0; i <= a.length; i++) {
        if(typeof a[i] == Lion)
            return LionLegCount(a[i]);
        if(typeof a[i] == Mouse)
            return MouseLegCount(a[i]);
        if(typeof a[i] == Snake)
            return SnakeLegCount(a[i]);
    }
}

AnimalLegCount(animals);
```

*Рисунок 2.2.3 – приклад коду*

Функція порушує принцип підстановки (і принцип відкритості-закритості). Цей код повинен знати про типи всіх оброблюваних їм об'єктів і, в залежності від типу, звертатися до відповідної функції для підрахунку кінцівок конкретного тваринного. Як результат, при створенні нового типу тваринного функцію доведеться переписувати:

```
//...
class Pigeon extends Animal {

}

const animals[]: Array<Animal> = [
    //...,
    new Pigeon();
]

function AnimalLegCount(a: Array<Animal>) {
    for(int i = 0; i <= a.length; i++) {
        if(typeof a[i] == Lion)
            return LionLegCount(a[i]);
        if(typeof a[i] == Mouse)
            return MouseLegCount(a[i]);
         if(typeof a[i] == Snake)
            return SnakeLegCount(a[i]);
        if(typeof a[i] == Pigeon)
            return PigeonLegCount(a[i]);
    }
}

AnimalLegCount(animals);
```

*Рисунок 2.2.3 – приклад коду*

Для того щоб ця функція не порушувала принцип підстановки, перетворимо її з використанням вимог, сформульованих Стівом Фентоном. Вони полягають в тому, що методи, які беруть або повертають значення з типом якогось суперкласу ( Animalв нашому випадку) повинні також приймати і повертати значення, типами яких є його підкласи ( Pigeon).

Озброївшись цими міркуваннями ми можемо переробити функцію AnimalLegCount:

```
function AnimalLegCount(a: Array<Animal>) {
    for(let i = 0; i <= a.length; i++) {
        a[i].LegCount();
    }
}

AnimalLegCount(animals);
```

*Рисунок 2.2.3 – приклад коду*

Тепер ця функція не цікавиться типами переданих їй об'єктів. Вона просто викликає їх методи LegCount. Все, що вона знає про типах - це те, що оброблювані їй об'єкти повинні належати класу Animalабо його подклассам.

Тепер в класі Animalповинен з'явитися метод LegCount:

```
class Animal {
    //...
    LegCount();
}
```

*Рисунок 2.2.3 – приклад коду*

А його подклассам потрібно реалізувати цей метод:

```
//...
class Lion extends Animal{
    //...
    LegCount() {
        //...
    }
}
//...
```

*Рисунок 2.2.3 – приклад коду*

В результаті, наприклад, при зверненні до методу LegCountдля екземпляра класу Lionздійснюється виклик методу, реалізованого в цьому класі, і повертається саме те, що можна очікувати від виклику подібного методу.

Тепер функції AnimalLegCountне потрібно знати про те, об'єкт якого саме підкласу класу Animalвона обробляє для того, щоб дізнатися відомості про кількість кінцівок у тварини, представленого цим об'єктом. Функція просто викликає метод LegCountкласу Animal, так як підкласи цього класу повинні реалізовувати цей метод для того, щоб їх можна було б використовувати замість нього, не порушуючи правильність роботи програми.

### 2.2.4 Принцип I: Interface Segregation Principle

Принцип поділу інтерфейсу

Цей принцип спрямований на усунення недоліків, пов'язаних з реалізацією великих інтерфейсів.

Він описує методи для малювання кіл ( drawCircle), квадратів ( drawSquare) і прямокутників ( drawRectangle). В результаті класи, що реалізують цей інтерфейс

і представляють окремі геометричні фігури, такі, як коло (Circle), квадрат (Square) і прямокутник (Rectangle), повинні містити реалізацію всіх цих методів. Виглядає це так:

Дивний у нас вийшов код. Наприклад, клас Rectangle, що представляє прямокутник, реалізує методи ( drawCirclei drawSquare), які йому абсолютно не потрібні. Те ж саме можна помітити і при аналізі коду двох інших класів.

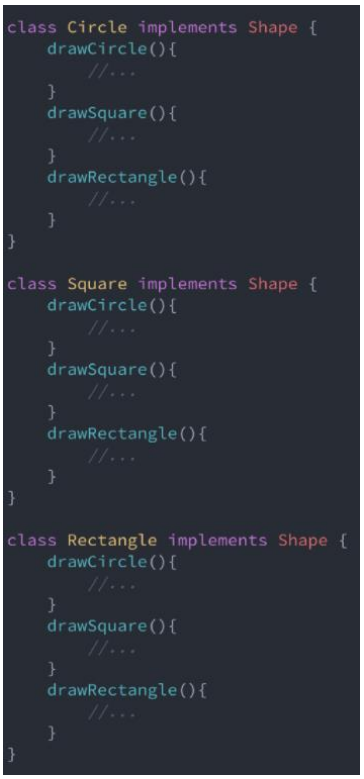



*Рисунок 2.2.3 – приклад коду*

Припустимо, ми вирішимо додати в інтерфейс Shapeще один метод, drawTrianglepризначений для малювання трикутників

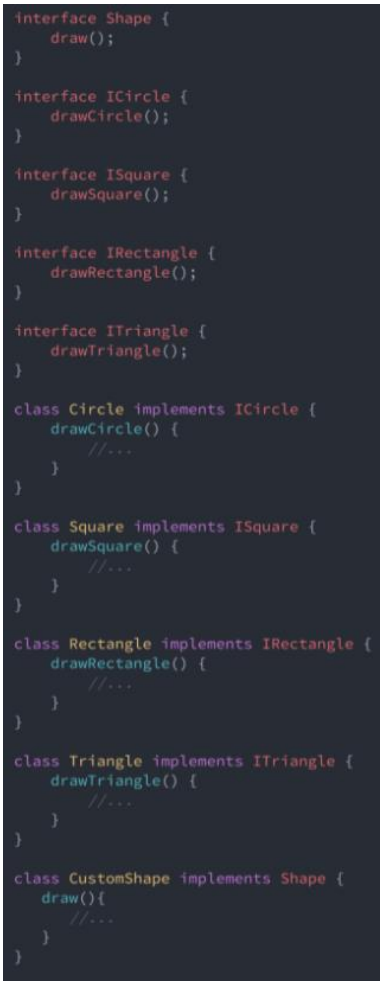
Це призведе до того, що класів, які представляють конкретні геометричні фігури, доведеться реалізовувати ще і метод drawTriangle. В іншому випадку виникне помилка.

Як видно, при такому підході неможливо створити клас, який реалізує метод для виведення кола, але не реалізує методи для виведення квадрата, прямокутника

і трикутника. Такі методи можна реалізувати так, щоб при їх виведенні викидалася б помилка, яка вказує на те, що подібну операцію виконати неможливо.

Принцип поділу інтерфейсу застерігає нас від створення інтерфейсів, подібних Shapez нашого прикладу. Клієнти (у нас це класи Circle, Squarei Rectangle) не повинні реалізовувати методи, які їм не потрібно використовувати. Крім того, цей принцип вказує на те, що інтерфейс повинен вирішувати лише якусь одну задачу (в цьому він схожий на принцип єдиної відповідальності), тому все, що виходить за рамки цього завдання, має бути винесено в інший інтерфейс або інтерфейси.

У нашому ж випадку інтерфейс Shapевирішує завдання, для вирішення яких необхідно створити окремі інтерфейси. Дотримуючись цієї ідеї, переробимо код, створивши окремі інтерфейси для вирішення різних вузькоспеціалізованих



```
interface Shape {
    draw();
}

interface ICircle {
    drawCircle();
}

interface ISquare {
    drawSquare();
}

interface IRectangle {
    drawRectangle();
}

interface ITriangle {
    drawTriangle();
}

class Circle implements ICircle {
    drawCircle() {
        //...
    }
}

class Square implements ISquare {
    drawSquare() {
        //...
    }
}

class Rectangle implements IRectangle {
    drawRectangle() {
        //...
    }
}

class Triangle implements ITriangle {
    drawTriangle() {
        //...
    }
}

class CustomShape implements Shape {
   draw(){
      //...
   }
}
```

завдань:

*Рисунок 2.2.5 – приклад коду*

Тепер інтерфейс ICirclевикористовується лише для малювання кіл, так само як і інші спеціалізовані інтерфейси - для малювання інших фігур. Інтерфейс Shapeможе застосовуватися в якості універсального інтерфейсу.

### 2.2.5 Принцип D: Dependency Inversion Principle

Принцип інверсії залежностей

1. Модулі верхніх рівнів не повинні залежати від модулів нижніх рівнів. Обидва типи модулів повинні залежати від абстракцій.
2. Абстракції не повинні залежати від деталей. Деталі повинні залежати від абстракцій.

В процесі розробки програмного забезпечення існує момент, коли функціонал додатка перестає поміщатися в рамках одного модуля. Коли це відбувається, нам доводиться вирішувати проблему залежностей модулів. В результаті, наприклад, може виявитися так, що високорівневі компоненти залежать від низькорівневих компонентів.


*Рисунок 2.2.5 – приклад коду*

Тут клас Httpє високорівнева компонент, а XMLHttpService-низькорівневий. Така архітектура порушує пункт А принципу інверсії залежностей: «Модулі верхніх рівнів не повинні залежати від модулів нижніх рівнів. Обидва типи модулів повинні залежати від абстракцій ».

Клас Httpвимушено залежить від класу XMLHttpService. Якщо ми вирішимо змінити механізм, який використовується класом Httpдля взаємодії з мережею - скажімо, це буде Node.js-сервіс або, наприклад, сервіс-заглушка, застосовуваний для цілей тестування, нам доведеться відредагувати всі екземпляри класу Http, змінивши відповідний код. Це порушує принцип відкритості-закритості.

Клас Httpне повинен знати про те, що саме використовується для організації мережевого з'єднання. Тому ми створимо інтерфейс Connection:

```
interface Connection {
    request(url: string, opts:any);
}
```

*Рисунок 2.2.5 – приклад коду*

Інтерфейс Connectionмістить опис методу requesti ми передаємо класу Httpаргумент типу Connection:

```
class Http {
    constructor(private httpConnection: Connection) { }
    get(url: string , options: any) {
        this.httpConnection.request(url,'GET');
    }
    post() {
        this.httpConnection.request(url,'POST');
    }
    //...
}
```

*Рисунок 2.2.3 – приклад коду*

Тепер, незалежно від того, що саме використовується для організації взаємодії з мережею, клас Httpможе користуватися тим, що йому передали, не піклуючись про те, що ховається за інтерфейсом Connection.

Перепишемо клас XMLHttpServicetаким чином, щоб він реалізовував цей інтерфейс:

```
class XMLHttpService implements Connection {
    const xhr = new XMLHttpRequest();
    //...
    request(url: string, opts:any) {
        xhr.open();
        xhr.send();
    }
}
```

*Рисунок 2.2.5 – приклад коду*

В результаті ми можемо створити безліч класів, що реалізують інтерфейс Connectioni підходять для використання в класі Httpдля організації обміну даними по мережі:

```
class NodeHttpService implements Connection {
    request(url: string, opts:any) {
        //...
    }
}

class MockHttpService implements Connection {
    request(url: string, opts:any) {
        //...
    }
}
```

*Рисунок 2.2.5 – приклад коду*

Як можна помітити, тут високорівневі і низькорівневі модулі залежать від абстракцій. Клас Http(високорівнева модуль) залежить від

інтерфейсу Connection(абстракція). Класи XMLHttpService, NodeHttpServicei MockHttpService(низькорівневі модулі) також залежать від інтерфейсу Connection.

Крім того, варто відзначити, що дотримуючись принципу інверсії залежностей, ми дотримуємося і принцип підстановки Барбари Лісков. А саме, виявляється, що типи XMLHttpService, NodeHttpServicei MockHttpServiceможуть служити заміною базового типу Connection.

# 3 РЕАЛІЗАЦІЯ СИСТЕМИ

## 3.1 Інструменти реалізації

Виходячи з поставленого завдання був обраний наступний стек технологій та інструментів:

— PHP 7, Yii2, Codeception - реалізація основної логіки;

— Python - для машинного навчання;

— NodeJS - comet server для гнізд;

— PostgreSQL - основна база даних;

— MongoDB, Cassandra, Google BigQuery - БД для аналітики;

— Redis - кеш і сесії;

— ElasticSearch - повнотекстовий пошук;

— RabbitMQ, Kafka - черги і message bus для аналітики;

— InfluxDB + Grafana - метрики;

— Graylog2, Zabbix - логирование і моніторинг системи;

— GitLab, Kubernetes, Docker, CloudFlare;

## 3.2 Реалізація основних функцій

Реалізація основних функцій веб-порталу в поданні призначених для користувача інтерфейсів представлена нижче на скріншотах з розробленої системи. Як приклад в Додатку Б представлений код контролерів для вибору треку і для логування.

Оскільки система є закритою, користувач спочатку потрапляє на форми реєстрації та авторизації (Рисунок 3.2. і Рисунок 3.2.1), де відповідно вводить свої облікові дані або проходить процес реєстрації.

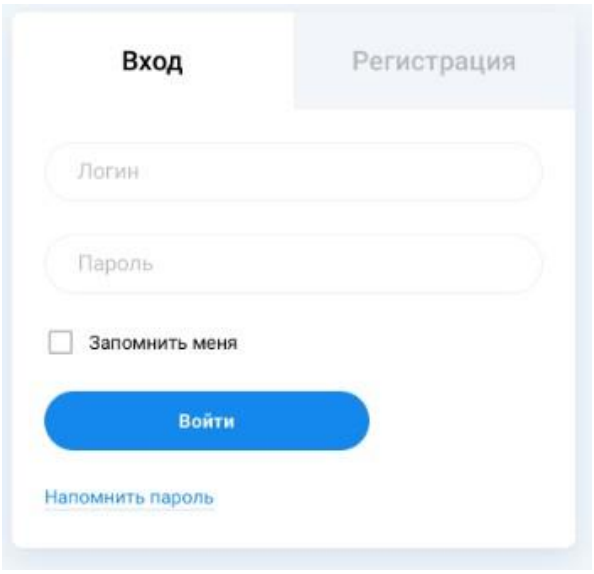



*Рисунок 3.2. - форма авторизації*

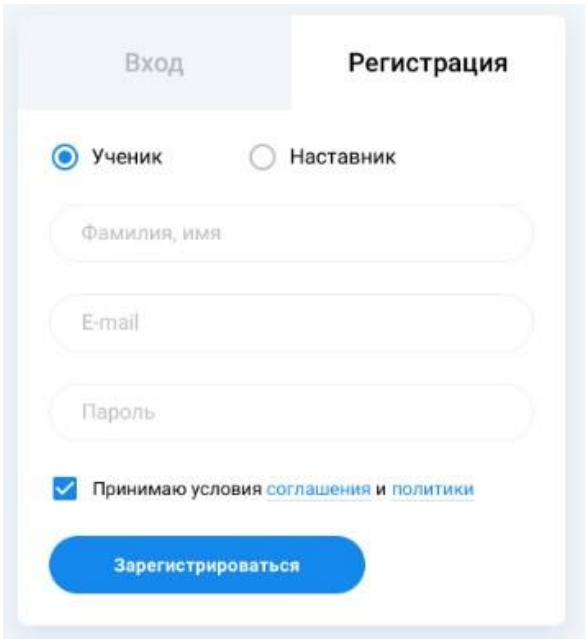



*Рисунок 3.2.1 - форма реєстрації*

Далі в залежності від своєї ролі в системі користувач бачить основний інтерфейс. Для ролі «учень», при першому вході в систему, на підставі розробленої методології, пропонується вибрати один з 5 способів вибору треку

індивідуального розвитку. Залежно від зробленого вибору система змінює вміст розділів.

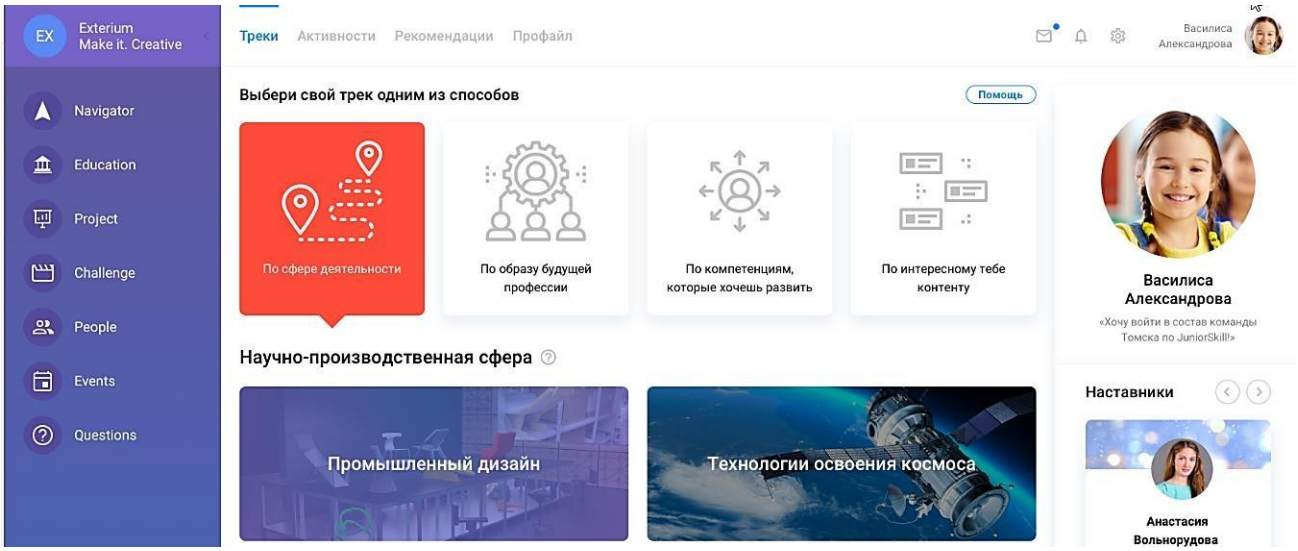



*Рисунок 3.2.2 - вибір треку*

Користувач може перейти на картку зацікавив треку і отримати більш повну інформацію про те, які можливості він надає. Коли користувач вибирає той чи інший трек, система перебудовує контентне зміст розділів порталу. Після проходження самооцінки (або інших варіантів придбання первинних компетенції) в профілі учня (рисунок 3.2.3) формується інтерактивна карта компетенцій (рисунок 3.2.4) окремо для базових і професійних компетенцій Додаток А2.

Залежно від прогресу з того чи іншого напрямку відбувається зміна зіркового діаграми і відповідно частки на круговій діаграмі.

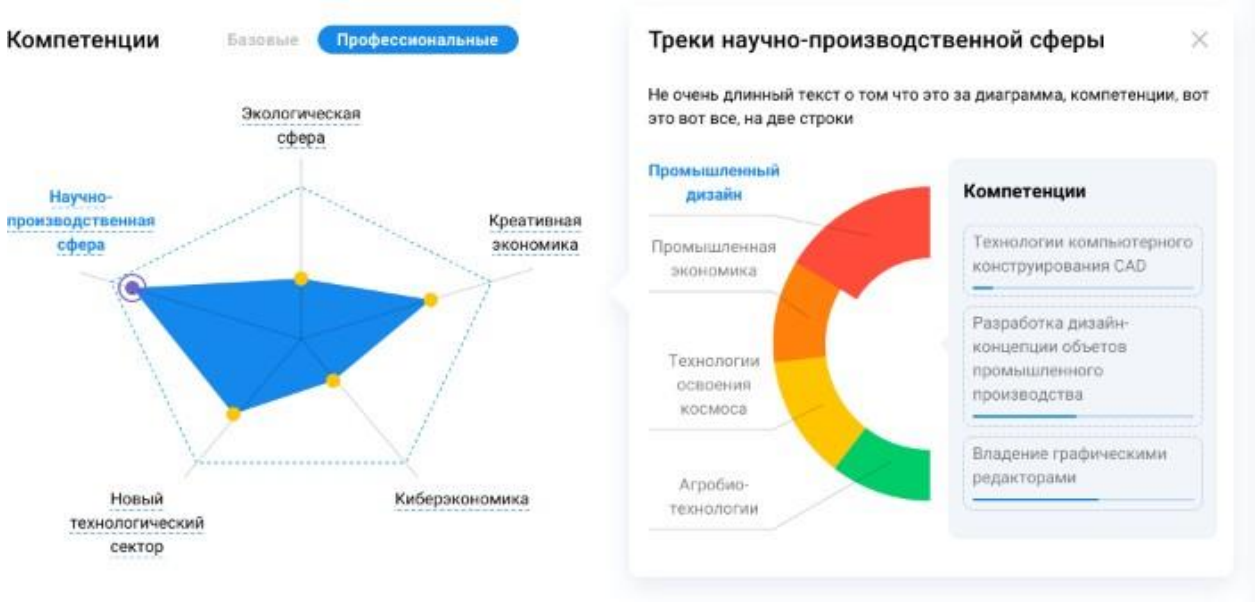



*Рисунок 3.2.3 - інтерактивна карта професійних компетенцій*

В розрізі кожної компетенції у користувача є можливість побачити матрицю цільових показників для її розвитку ( Рисунок 3.2.6 Додаток А3). Відповідно при проходженні освітніх активностей на порталі або поза ним відбувається нарахування балів і прогрес у розвитку компетенцій.

При великій кількості цікавого контенту на порталі є ризик, що допитливі учні виберуть занадто багато освітніх активностей, тому в профілі учня передбачений інструмент, що дозволяє наставникам, батькам і дітям контролювати рівень завантаження учня ( Рисунок 3.2.7 Додаток А). Також на цьому малюнку можна побачити реалізацію обліку особистих досягнень учня, його захоплень і інших навичок. На вкладці «Активності», яка є центральною в процесі споживання освітнього контенту, користувачеві доступна інформація про його просуванні по треку в формі таймлайна, поточних завданнях по курсам (в тому числі статуси), розкладі занять, освітніх активностях в яких він бере участь (або подав заявку на участь), а також архіви його активностей ( Рисунок 3.2.8 Додаток А3).

Після завершення кожної освітньої активності система проходить за алгоритмом вироблення рекомендації, розглянутому раніше. В результаті відображення контенту на платформі змінюється ( Рисунок 3.2.9 і Рисунок 3.2.10).

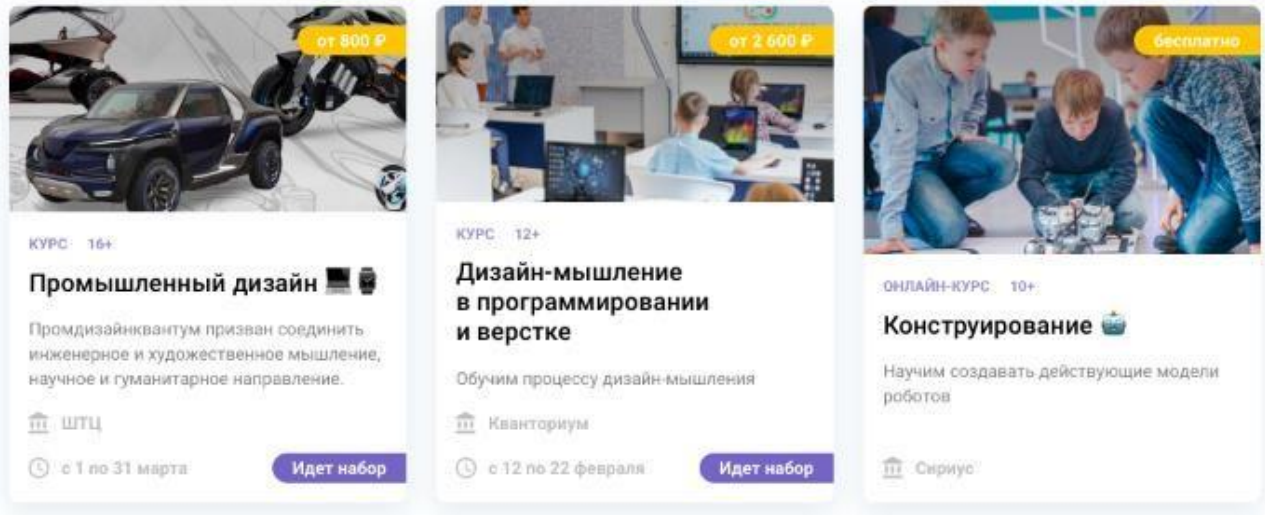



*Рисунок 3.2.4 - Приклад реалізації зміни контенту до проходження курсу*

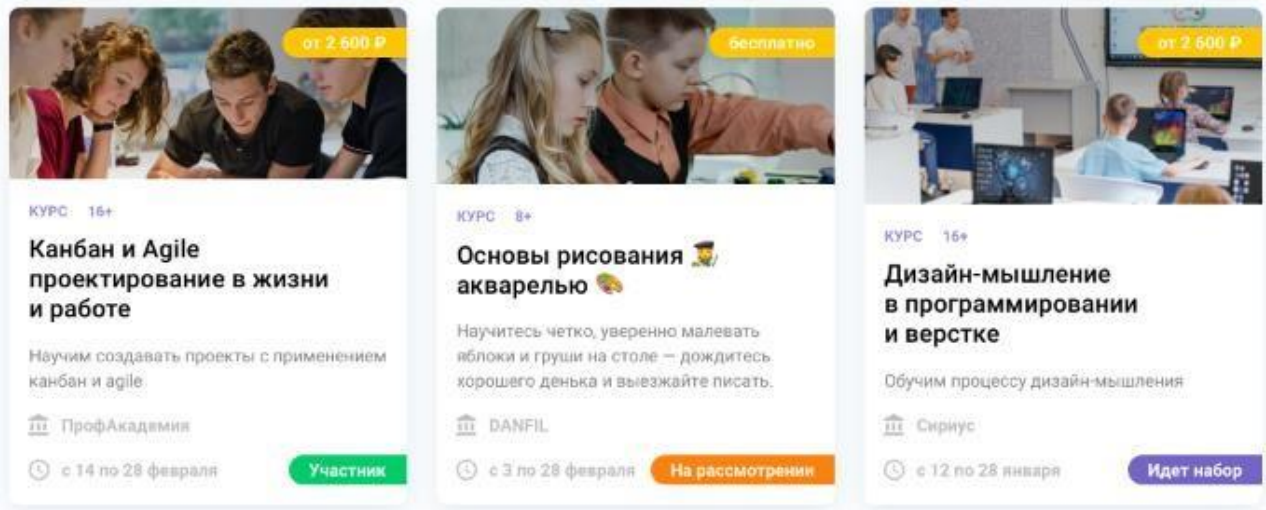



*Рисунок 3.2.5 - Приклад реалізації зміни контенту після проходження курсу*

Таким чином, на етапі розробки системи були реалізовані основні сценарії використання системи для кожної з передбачених ролей користувачів, відповідно

до вимог технічного завдання, структурою і логічними взаємозв'язками сутностей і в рамках розробленої методології.

У першій версії додатка користувач має можливість здійснити вибір одного з 5 способів побудови освітньої траєкторії: за компетенціями, за освітнім контенту, по треку розвитку, за образом майбутньої професії, через оцінку наставником.

За результатами можемо зробити висновок, про те, що готовність проекту до комерціалізації середня. Необхідно залучити компетентних фахівців в команду розробки, опрацювати механізми реалізації з усіма зацікавленими сторонами, а також вивчити питання виходу на закордонні ринки.

### 3.3 Цілі і результат проекту.

В даному розділі необхідно привести інформацію про зацікавлених сторонах проекту, ієрархії цілей проекту і критерії досягнення цілей.

Таблиця 3.3 - Цілі і результат проекту

| | |
|---|---|
| **Мета проекту:** | Створити освітню платформу додаткового освіти шко лярів |
| **Очікувані результати проекту:** | Готова і функціонує освітня платформа |
| **Критерії приймання результату проекту:** | Технічне завдання |

### 3.4 Ієрархічна структура роботи проекту

*Ієрархічна структура робіт* (ІСР) - деталізація укрупненої структури робіт.

У процесі створення ІСР структурується і визначається зміст всього проекту. В рамках даного проекту була обрана каскадна модель управління проектом, яка має наувазі послідовне проходження стадій від збору вимог і до експлуатації та підтр имки проекту. Остання стадія виключена в виду того, що проект після тестуванн я буде переданий на підтримку замовнику.

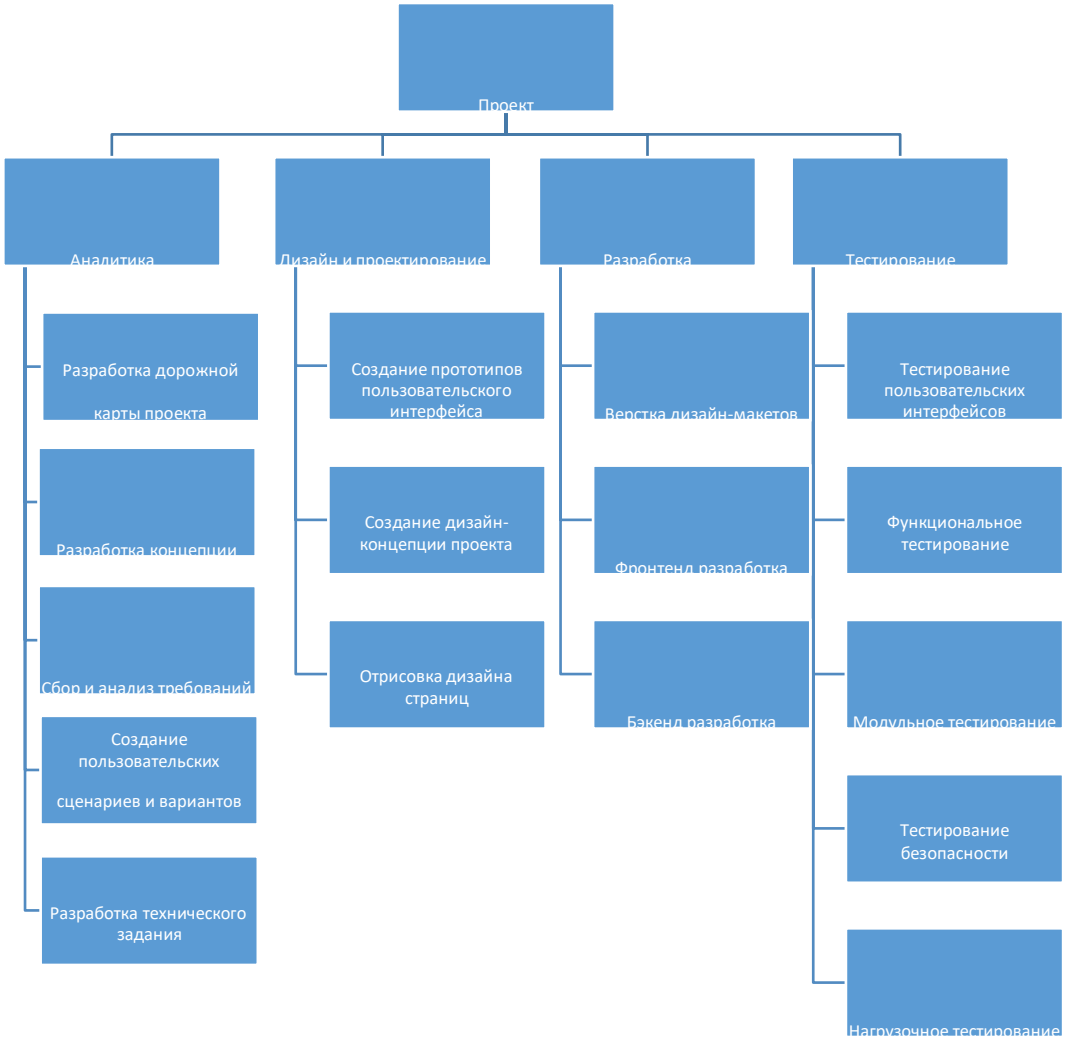



*Рисунок 3.4. - Ієрархічна структура робіт*

## ВИСНОВОК

За результатами виконання дипломної роботи був розроблений веб-портал додаткової освіти школярів.

В ході роботи було проведено аналіз предметної області та теоретичних основ розробки освітніх веб-порталів в результаті, якого були виявлені характеристики та основні функції, якими повинен володіти сучасний освітній портал. Зокрема, важливою характеристикою є персоналізація контенту для кінцевих користувачів. В системі це реалізовано завдяки механізму адаптації контенту в залежності від інтересів учня.

Проведено аналіз конкурентних систем на підставі якого був зроблений висновок про те, що систем спрямованих на додаткову освіту школярів, які дозволяють вибудовувати індивідуальну траєкторію розвитку учня (відповідно до його схильностями і інтересами) з прицілом на реальну проектну та професійну діяльність, немає.

Базуючись на обраному архітектурному патерні MVP було описано структуру

класів програмного забезпечення. Розроблено ефективний алгоритм створення заявки, на основі якого було розроблено клієнтський застосунок

Складено портрети потенційних споживачів, на основі яких розроблені ролі в системі і побудована значна частина концептуальної аналітики.

Проведена вартісна і часова оцінка проекту, складений календарний план проекту у вигляді діаграми Гантта.

Серед усього різноманіття методологій розробки було здійснено вибір спіральної моделі, яка відноситься до ітеративним методологій, як найбільш підходящий під завдання, обмеження і ризики проекту.

Розроблено методику побудови освітніх траєкторій, яка лягла в основу логіки роботи системи.

Проведено збір та аналіз вимог до системи широким спектром методів: інтерв'ю та опитування, мозкові штурми, прототипування, вивчення аналогів.

Здійснено вибір підходящої під завдання архітектури, а саме сервіс-орієнтованої архітектури, що дозволяє реалізовувати незалежні і взаємозамінні мікросервіси на різних мовах програмування. Даний підхід хоча і вимагає істотних початкових витрат, але при розширенні системи дозволяє заощадити великий обсяг фінансових коштів. А також стає можливим спростити процес масштабування системи, який передбачається на наступних ітераціях розробки.

Описано основні сценарії використання системи і поля сутностей бази даних. Складено технічне завдання за модифікованим шаблоном Симкина.

Обрані інструменти реалізації системи і відповідний технологічний стек, який дозволяє звести воєдино і вирішити спектр різнопланових завдань проекту, в тому числі з доробком на наступні етапи розробки.

В результаті розробки вийшов сучасний освітній портал з унікальним функціоналом, завдяки покладеної в основу методики побудови освітніх траєкторій.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Бабаєв, А.Г. Створення сайтів [Текст] О.Г.. Бабаєв. СПб .: Пітер, 2013.
2. Бужинська, Н.В., Васильєв, Р.Ю. Основні етапи проектування сайтів // Сучасні технології в світовому науковому просторі. Збірник статей Міжнародної науково-практичної конференції 25 січня 2016р., Томськ, 2016. С. 18-20.
3. Ігнатова, Н.Г. Інтернет-технології в системі освіти [Текст] / Н.Г. Ігнатова. М .: Прес 2009.
4. Клименко, Р.М. Веб-мастеринг [Текст] / Р.М. Клименко. М., 2010. Пунина, Т.Д. Проектування і розміщення в мережі Інтернет адміністративних сайтів [Текст] / Т.Д. Пунина. М., 2011 року.
5. Хуторський, А. В. Педагогічна інноватика [Текст] / А.В. Хуторський. М .: Изд-во УНЦ ДО 2009.
6. Дочкиного, С. А. Інформатизація додаткової професійної освіти професійно-педагогічних кадрів: організаційно педагогічний аспект [Текст] / С. А. дочкиного: монографія. - СПб .: Арден, 2010. - 226 с.
7. Освітній портал [Електронний ресурс] / Гос.НІІ ІТ і ТК «Інформіка». - Режим доступу: http://tm.ifmo.ru/db/doc/presentation, вільний.
8. ГОСТ 19.201-78 Єдина система програмної документації. Технічне завдання. Вимоги до змісту та оформлення.
9. ГОСТ 34.602-89 Інформаційна технологія. Комплекс стандартів на автоматизовані системи.
10. Технічне завдання на створення автоматизованої системи. РД 50-34.698-90 Автоматизовані системи. Вимоги до змісту документів
11. Вігерс Карл «Розробка вимог до програмного забезпечення» / Пер, з англ. - М .: Іздательсш-торговий дім «Російська Редакція», 2004. -576с.
12. Макнейл Патрік Веб-дизайн. Книга ідей веб-розробника; Пітер - Москва, 2014. - 288 с.

13. Метью Девід HTML5. Розробка веб-додатків; Рід Груп - Москва, 2012. - 320 с.
14. Хасслер Марк Веб-аналітика; Ексмо - Москва, 2010. - 432 с.
15. Хоган Б., Уоррен К., Вебер М., Джонсон К., Годін А. Книга веб- програміста. Секрети професійної розробки веб-сайтів; Пітер - Москва, 2013. - 288 с.
16. Шкляр Леон, Розен Річ Архітектура веб-додатків; Ексмо - Москва, 2011. - 640 с.
17. Коберн А. Сучасні методи опису функціональних вимог до систем. - М .: Лорі, 2002. Левенчук А. І. Системне мислення. Підручник. - Вид-во «Видавничі рішення». - 2018 г. - 398 с.
18. Іванніков А.Д., Тихонов А.Н. Основні положення концепції створення систем освітніх порталів / Інтернет-портали. Зміст і технології. Збірник наукових статей. Випуск 1. - Москва, Просвещение, 2003. - с. 8-18.
19. Булгаков М.В., Внотченко С.С., Грідіна Є.Г. Реалізація каталогу освітніх Інтернет-ресурсів федерального порталу "Острів знань" / Інтернет-портали. Зміст і технології. Збірник наукових статей. Випуск 1. - Москва, Просвещение, 2003. - с. 460-497.
20. Алексєєв В.В., Грідіна Є.Г., Корольов П.Г. Куракіна Н.І. Оцінка якості освітніх Інтернет-ресурсів. Збірник праць III Міжнародній науково-методичної конференції "Системи управління якістю вищої освіти", - Воронеж, 2003.

# ДОДАТКИ

Додаток А

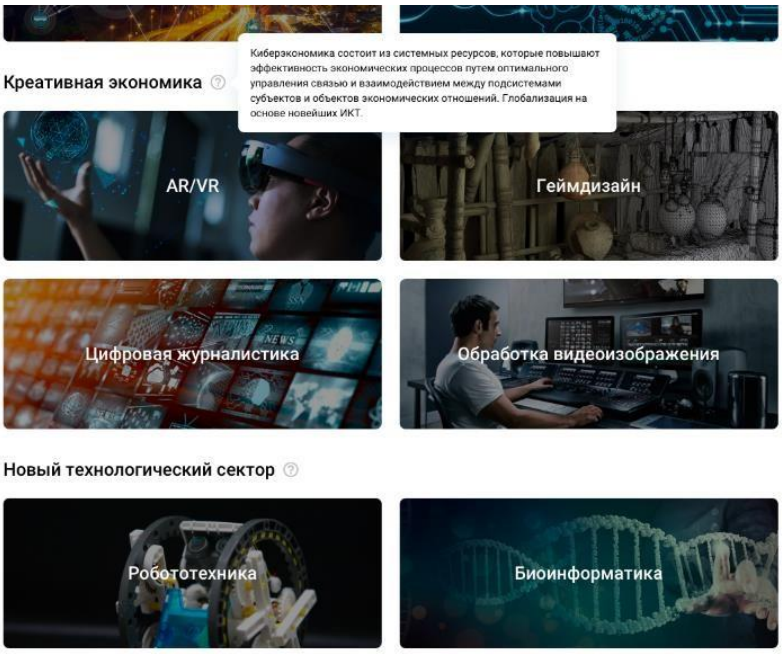



*Рисунок 3.2.3 – доступні на вибір треки*

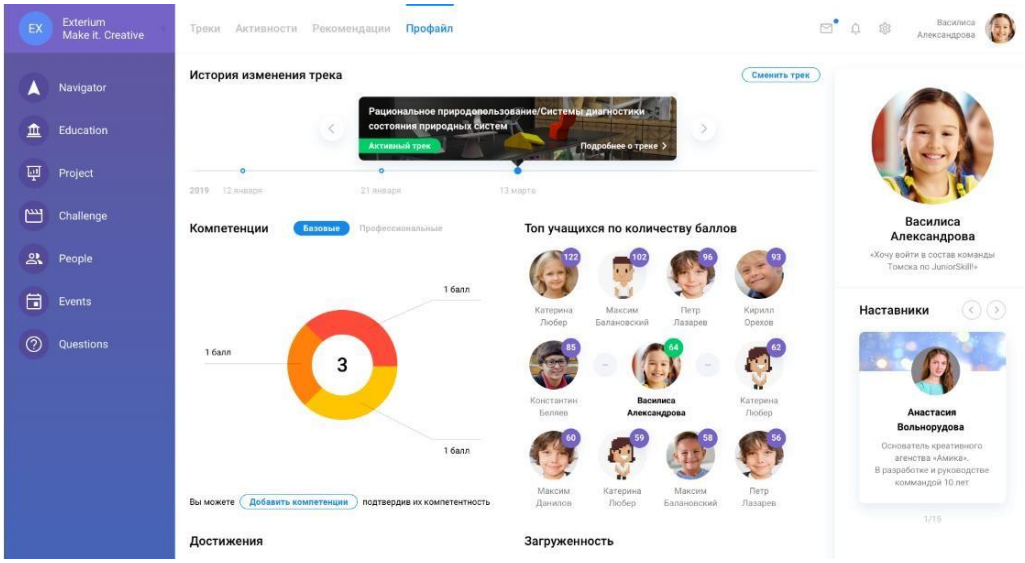


*Рисунок 3.2.4 – профіль користувача*

Додаток Б

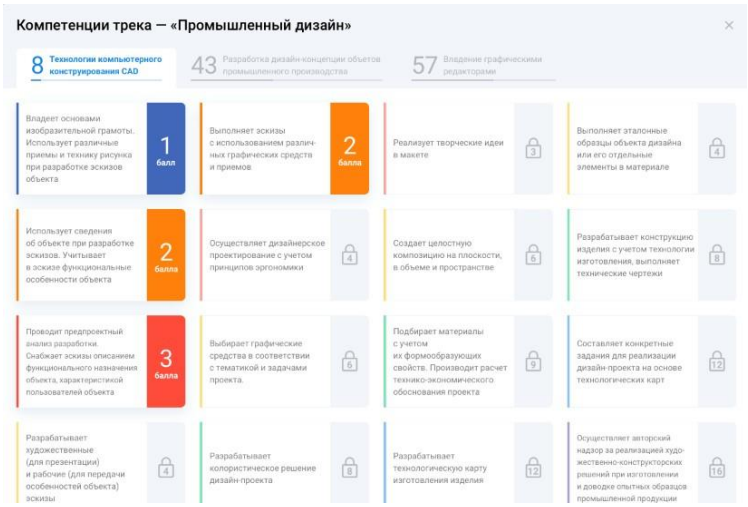



*Рисунок 3.2.6 – реалізація матриці цільових показників формування компетенції*

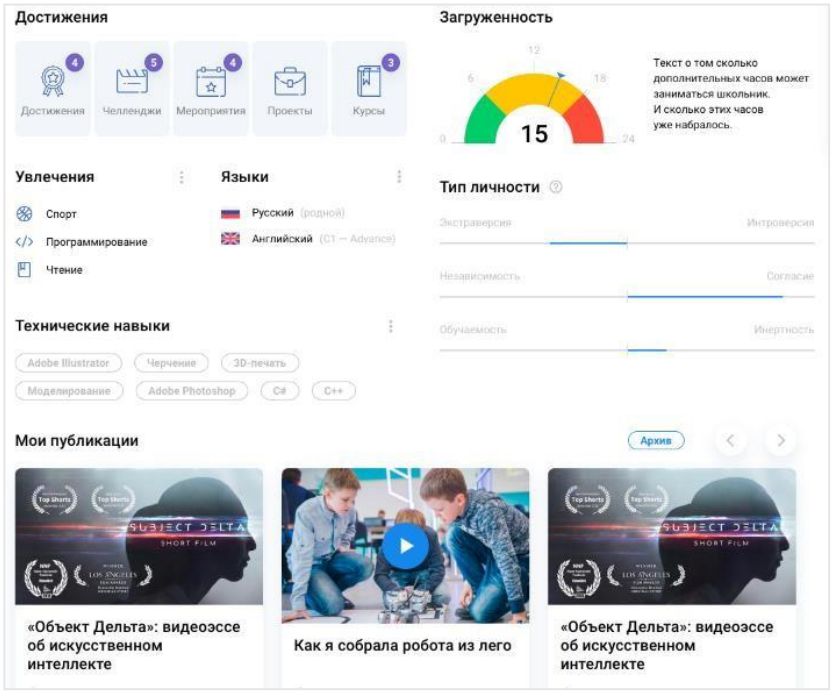



*Рисунок 3.2.7 – продовження сторінки профілю учня*

Додаток Б

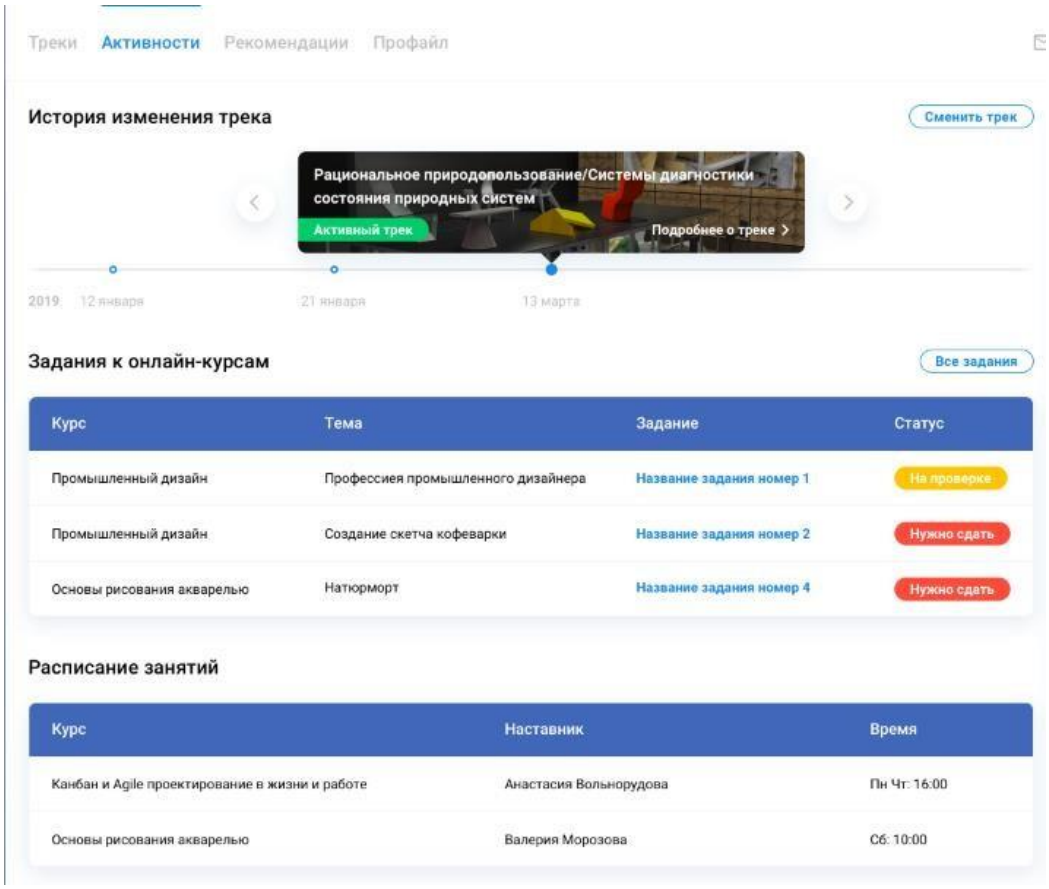



*Рисунок 3.2.8 – Частина сторінки активності*

Додаток В

## Приклад коду контролерів

```php
<?php
namespace backend\controllers;

use backend\components\BaseController;
use backend\models\search\SubscribeSearch;
use backend\models\Subscribe;
use Yii;
use yii\helpers\Url;
use yii\web\NotFoundHttpException;

class SubscribeController extends BaseController
{
    /**
     * @var $modelClass \yii\db\ActiveRecord
     * @var $searchModelClass \yii\db\ActiveRecord
     */
    public $modelClass = Subscribe::class;
    public $searchModelClass = SubscribeSearch::class;
    public $newModelDefaultAttributes = ['status' =>
Subscribe::STATUS_ACTIVE];

    /**
     * @param $id
     * @return string|\yii\web\Response
     * @throws NotFoundHttpException
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
        $model->tagList = $model->tags;
        $model->sectionList = $model->sections;

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(Url::previous());
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

<?php

namespace backend\controller
```

```
use backend\models\search\LogSearch;
use Yii;

class LogController extends \yii\web\Controller
{
    public function actionIndex()
    {
        $searchModel = new LogSearch();
        $dataProvider = $searchModel->search(Yii::$app->request-
>queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
}
```