ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА ДОДАТКУ 'EASY LEARN' З ВИКОРИСТАННЯМ REACT NATIVE»**

Виконала: студентка 4 курсу, групи ПД-44

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Рудська А.І.

(прізвище та ініціали)

Керівник Гребенюк В.В.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного

забезпечення

_____О.В. Негоденко

«____»_____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ
РУДСЬКІЙ АНАСТАСІЇ ІГОРІВНІ

1.Тема роботи: «Розробка додатку 'EasyLearn’ з використанням React Native»
Керівник роботи Гребенюк Віктор Вікторович, доцент кафедри
затверджені наказом вищого навчального закладу від — «12» березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вхідні дані до роботи:

3.1. Середовище розробки Visual Studio 2017 Professional

3.2. Алгоритм роботи додатку

3.3. Udemy API, Prometheus API

3.4. Науково-технічна література, пов’язана з розробкою мобільного додатку на основі API

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

4.1. Аналіз обов’язків розроблюваного додатку

4.2. Аналіз та порівняння існуючих прототипів

4.3. Дослідження програмних засобів для розробки додатку

4.4. Розробити функціонал створеного додатку

5. Перелік графічного матеріалу

5.1.1. Популярність освітніх додатків

5.1.2. Переваги та недоліки найвідоміших освітніх додатків

5.1.3. Програмні засоби реалізації

5.1.4. Огляд можливостей розробленого додатку

5.1.5. Апробація результатів досліджень

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	14.03.21 – 24.03.21	
2	Аналіз існуючих прототипів	24.03.21 – 25.03.21	
3	Дослідження програмних засобів	25.03.21 – 02.04.21	
4	Моделювання об’єкту проектування	02.04.21 – 13.04.21	
5	Розробка функціоналу	13.04.21 – 29.04.21	
6	Вступ, висновки, реферат	29.04.21 – 08.05.21	
7	Розробка презентації застосунку	08.05.21 – 24.05.21	
8	Попередній захист роботи	25.05.21	

Студент _____ _____

(підпис) (прізвище та ініціали)

Керівник роботи _____ _____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 42с., 7 рис., 18 джерел.

Ключеві слова: Visual Studio, React Native, Android, EasyLearn, API, мобільний додаток, навчальні матеріали.

Об'єкт дослідження – процес пошуку матеріалів для навчання.

Предмет дослідження – мобільний додаток, для пошуку необхідних навчальних матеріалів.

Мета роботи – розробка мобільного додатку для зручності пошуку необхідних навчальних курсів.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

Наукова новизна даної роботи полягає в наступному:

1. Розроблено алгоритм для полегшення та зручності придбання навчальних матеріалів.
2. Визначено, що додаток який розроблений на основі API з використанням React Native, показує не погані результати по швидкодії.
3. На основі результатів виконаних порівнянь та досліджень, був розроблений додаток “EasyLearn”.

В роботі виконано аналіз існуючих додатків аналогів. Встановлено переваги та недоліки існуючих додатків. В результаті аналізу було визначено основні потреби користувачів. Проаналізовано можливості середовища розробки Visual Studio 2017 Professional. Розроблено логіку практичних завдань та загальну концепцію представлення інформації для користувачів.

Галузь використання – мобільний додаток може використовувати будь-яка людина, яка має доступ до інтернету та акаунт у додатку Play Market.

ЗМІСТ

ВСТУП

Інтернет-освіта — це прогресивна форма навчання, що здійснюється за допомогою ресурсів та технологій мережі інтернет. Можливість доступу в Інтернет майже з будь-якого місця кардинально змінила спосіб обробки і отримання інформації. Онлайн навчання має багато переваг, одна з яких полягає в можливості навчатися з дому або з будь-якого іншого місця.

З комп'ютера, планшета або телефону можливо не тільки одноразово поповнювати свій багаж знань у різних сферах діяльності, але і розвивати себе в конкретній галузі.

Дистанційне навчання має багато технічних можливостей. Таких, як відеороліки, аудіо-подкасти, онлайн-тестування.

Існує безліч різних сайтів і додатків, націлених на здобуття нових знань у різних сферах діяльності. Наприклад, сайти з освітніми матеріалами, Coursera.org, Udemy.com або Dou.ua, це крупні провайдери відкритих онлайн-курсів. Вони працюють з університетами та іншими організаціями, пропонуючи онлайн-курси та сертифікати з різноманітних галузей.

Метою даної роботи є розробка мобільного додатка для зручності пошуку необхідних освітніх матеріалів. Додаток дозволяє швидко знаходити потрібний матеріал, є можливість ознайомитись із планом, тривалістю та вартістю курсу, дозволяє додати до обраного курс який сподобався, а також ділиться посиланням на сайт з освітніми матеріалами для придбання обраного онлайн-курсу.

Для реалізації програми було виконано аналіз існуючих додатків, встановлено їх переваги та недоліки, розглянуто дизайн інтерфейсу, проаналізовано можливості та розроблено функціональність додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Мобільні додатки

Мобільний додаток, який також називають мобільним додатком або просто додатком, - це комп’ютерна програма або програмне забезпечення, призначене для роботи на мобільному пристрої, такому як телефон, планшет або годинник. Програми спочатку були призначені для сприяння підвищенню продуктивності, наприклад електронної пошти, календаря та контактних баз даних, але суспільний попит на програми спричинив швидке розширення в інших сферах, таких як мобільні ігри, автоматизація роботи на заводі, послуги GPS та локації, відстеження замовлень та квитки покупки, завдяки чому зараз доступні мільйони програм. Зазвичай програми завантажуються з платформ розповсюдження програм, якими керує власник мобільної операційної системи, наприклад, App Store (iOS) або Google Play Store. Деякі додатки безкоштовні, а інші мають ціну, при цьому прибуток розподіляється між творцем програми та платформою розповсюдження. Мобільні програми часто відрізняються від настільних програм, призначених для роботи на настільних комп’ютерах, та веб-програм, які працюють у мобільних веб-браузерах, а не безпосередньо на мобільному пристрої.

У 2009 році оглядач технологій Девід Пог заявив, що смартфони можуть отримати прізвисько "телефонні додатки", щоб відрізнити їх від раніше менш складних смартфонів. Термін "додаток", скорочене від "програмне забезпечення", з тих пір став дуже популярним; у 2010 році Американське товариство діалектів визначило його "Словом року".

Більшість мобільних пристроїв продаються з декількома програмами, що входять до складу попередньо встановленого програмного забезпечення, наприклад, веб-браузером, поштовим клієнтом, календарем, картографічною програмою та додатком для придбання музики, інших засобів масової інформації чи інших додатків. Деякі попередньо встановлені програми можна видалити

звичайним процесом видалення, залишаючи таким чином більше місця для бажаних. Якщо програмне забезпечення не дозволяє цього, деякі пристрої можуть використовуватися для усунення небажаних програм.

Програми, які не встановлені заздалегідь, зазвичай доступні через платформи розповсюдження, які називаються магазинами програм. Вони почали з'являтися в 2008 році, і ними, як правило, керує власник мобільної операційної системи, наприклад, Apple App Store, Google Play, Windows Phone Store та BlackBerry App World. Однак існують незалежні магазини додатків. Деякі програми безкоштовні, а інші потрібно купувати. Зазвичай вони завантажуються з платформи на цільовий пристрій, але іноді їх можна завантажити на ноутбуки або настільні комп'ютери. Для додатків із ціною, як правило, відсоток, 20-30%, надходить до постачальника дистрибуції (наприклад, iTunes), а решта надходить до виробника програми. Отже, той самий додаток може коштувати різну ціну залежно від мобільної платформи.

Програми також можна встановлювати вручну, наприклад, запускаючи пакет програм Android на пристроях Android.

Спочатку мобільні програми пропонувались для загальної продуктивності та пошуку інформації, включаючи електронну пошту, календар, контакти, фондовий ринок та інформацію про погоду. Однак суспільний попит та доступність інструментів для розробників спонукали до швидкого розширення на інші категорії, такі як ті, що обробляються настільними прикладними програмними пакетами. Як і у випадку з іншим програмним забезпеченням, вибух кількості та різноманітності програм зробив відкриття проблемою, що, в свою чергу, призвело до створення широкого спектру оглядів, рекомендацій та джерел інформації, включаючи блоги, журнали та спеціальні послуги з виявлення додатків в Інтернеті. . У 2014 році державні регуляторні органи почали намагатися регулювати та керувати програмами, зокрема медичними. Деякі компанії пропонують додатки як альтернативний спосіб доставки вмісту з певними перевагами перед офіційним веб-сайтом.

Зі збільшенням кількості мобільних додатків, доступних у магазинах додатків, і вдосконаленими можливостями смартфонів, люди завантажують більше додатків на свої пристрої. Використання мобільних додатків стає дедалі поширенішим серед користувачів мобільних телефонів. Дослідження comScore, проведене в травні 2012 року, повідомляло, що протягом попереднього кварталу більше абонентів мобільних пристроїв використовувало додатки, ніж переглядало веб-сторінки на своїх пристроях: 51,1% проти 49,8% відповідно. Дослідники виявили, що використання мобільних додатків суттєво корелює з контекстом користувача та залежить від місцезнаходження користувача та часу доби. Мобільні програми відіграють дедалі більшу роль у сфері охорони здоров'я, і коли їх правильно розробити та інтегрувати, це може принести багато переваг.

Фірма з досліджень ринку Gartner передбачила, що в 2013 році буде завантажено 102 мільярди додатків (91% з них безкоштовно), що принесе в США 26 мільярдів доларів, що на 44,4% більше порівняно з 18 мільярдами доларів США у 2012 році. До другого кварталу 2015 року лише магазини Google Play та Apple заробили 5 мільярдів доларів. За підрахунками аналітичного звіту, економіка додатків створює дохід на суму понад 10 млрд. Євро на рік в межах Європейського Союзу, тоді як у 28 штатах ЄС через зростання ринку додатків створено понад 529 000 робочих місць.

1.2 Розробка мобільних додатків

Розробка мобільних додатків - це процес або процес, за допомогою якого розробляється мобільний додаток для мобільних пристроїв, таких як особисті цифрові помічники, корпоративні цифрові помічники або мобільні телефони. Ці програми можна попередньо встановити на телефони під час виробничих платформ або доставити як веб-програми, використовуючи обробку на стороні сервера або клієнта (наприклад, JavaScript), щоб забезпечити "подібну до програми" роботу у веб-браузері. Розробники прикладного програмного забезпечення також повинні

враховувати довгий спектр розмірів екрану, технічних характеристик та конфігурацій через сильну конкуренцію в мобільному програмному забезпеченні та зміни на кожній з платформ. Розробка мобільних додатків неухильно зростає, збільшуються доходи та створюються робочі місця. Згідно з аналітичним звітом 2013 року, в ЄС налічується 529 000 робочих місць з економікою прямих додатків, тоді як 28 членів (включаючи Великобританію), 60 відсотків з яких є розробниками мобільних додатків.

Як частина процесу розробки, дизайн мобільного користувацького інтерфейсу (UI) також має важливе значення для створення мобільних додатків. Мобільний інтерфейс розглядає обмеження, контексти, екран, введення та мобільність як основи дизайну. Користувач часто є центром взаємодії зі своїм пристроєм, а інтерфейс включає компоненти як апаратного, так і програмного забезпечення. Введення користувачами дозволяє користувачам маніпулювати системою, а вихід пристрою дозволяє системі вказувати на ефекти маніпуляцій користувачів. Обмеження дизайну мобільного інтерфейсу включають обмежену увагу та форм-фактори, такі як розмір екрану мобільного пристрою для рук (рук) користувача. Контексти мобільного інтерфейсу сигналізують про підказки від активності користувачів, таких як розташування та планування, які можуть відображатися в результаті взаємодії користувачів у мобільному додатку. Загалом, мета дизайну мобільного інтерфейсу - головним чином зрозумілий, зручний інтерфейс. Інтерфейс мобільних додатків повинен: враховувати обмежену увагу користувачів, мінімізувати натискання клавіш та бути орієнтованим на завдання з мінімальним набором функцій. Цю функціональність підтримують мобільні корпоративні платформи додатків або інтегровані середовища розробки (IDE).

Мобільні інтерфейси або інтерфейсні інтерфейси покладаються на мобільні фонові інтерфейси для підтримки доступу до корпоративних систем. Мобільний сервер сприяє маршрутизації даних, безпеці, автентифікації, авторизації, роботі в режимі офлайн та організації служб. Ця функціональність підтримується поєднанням проміжних компонентів, включаючи сервер мобільних додатків,

мобільний серверний сервіс як послугу (MBaaS) та інфраструктуру орієнтованої на службу архітектури (SOA).

Розробка мобільних додатків стає все більш важливою для багатьох підприємств, у яких понад 3 мільярди людей у всьому світі використовують смартфони, понад 1,5 мільярда - планшети станом на 2019 рік. Користувачі в середньому проводять 90 відсотків свого мобільного часу в додатках, а їх більше 700 мільйони завантажень програм із різних магазинів програм.

1.3 Дистанційне навчання

Дистанційне навчання - це навчання учнів, які не завжди можуть бути фізично присутніми в школі. Традиційно це, як правило, передбачало заочні курси, на яких студент листувався зі школою поштою. Сьогодні це передбачає онлайн-освіту. Програма дистанційного навчання може бути повністю дистанційною, або поєднанням дистанційного навчання та традиційного навчання в класі. Масивні відкриті онлайн-курси (MOOC), що пропонують широкомасштабну інтерактивну участь та відкритий доступ через всесвітню павутину або інші мережеві технології, є останніми навчальними способами дистанційної освіти. Ряд інших термінів (розподілене навчання, електронне навчання, m-навчання, онлайн-навчання, віртуальний клас тощо) використовується приблизно синонімічно дистанційній освіті.

Одна з найперших спроб була рекламована в 1728 році. Це було у "Бостонській газеті" для "Калеба Філіпса, вчителя нового методу короткої руки", який шукав учнів, які бажали вчитися на щотижневих поштових уроках.

Перший курс дистанційного навчання в сучасному розумінні провів сер Ісаак Пітман у 1840-х роках, який викладав систему стенографії, розсилаючи тексти, переписані у стенограму на листівках, і отримуючи транскрипції від своїх учнів натомість для виправлення. Елемент зворотного зв'язку студентів був вирішальним

нововведенням у системі Пітмана. Ця схема стала можливою завдяки введенню єдиних ставок поштових відправлень по всій Англії в 1840 р.

Цей ранній початок виявився надзвичайно успішним, і через три роки було створено Товариство фонографічної кореспонденції для створення цих курсів на більш офіційній основі. Товариство відкрило шлях для пізнішого формування коледжів сера Ісаака Пітмана по всій країні.

Першою заочною школою в США було Товариство заохочення навчання вдома, яке було засноване в 1873 р.

Заснований в 1894 році, Оксфорд, Волсі Холл, був першим коледжем дистанційного навчання у Великобританії.

Пандемія COVID-19 призвела до закриття переважної більшості шкіл у всьому світі. Багато шкіл перейшли на дистанційне навчання через Інтернет за допомогою різноманітних платформ, включаючи Zoom, Cisco Webex, Google Classroom, Microsoft Teams, D2L та Edgenuity. Виникли занепокоєння щодо впливу цього переходу на студентів, які не мають доступу до пристрою з підтримкою Інтернету або стабільного з’єднання з Інтернетом.

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 JavaScript

JavaScript (/ ˈdʒɑːvəˌskrɪpt /), часто скорочується як JS, є мовою програмування, що відповідає специфікації ECMAScript. JavaScript є високорівневим, часто вчасно скомпільованим і багатопарадигмальним. Він має синтаксис фігурних дужок, динамічне введення тексту, орієнтацію на об’єкти на основі прототипу та функції першого класу.

Поряд з HTML та CSS, JavaScript є однією з основних технологій Всесвітньої мережі. Понад 97% веб-сайтів використовують його на стороні клієнта для поведінки веб-сторінок, часто включаючи сторонні бібліотеки. Усі основні веб-браузери мають спеціальний механізм JavaScript для виконання коду на пристрої користувача.

Як мова багатопарадигми, JavaScript підтримує керовані подіями, функціональні та імперативні стилі програмування. Він має інтерфейси програмування програм (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних та об’єктною моделлю документа (DOM).

Стандарт ECMAScript не включає введення / виведення, наприклад, мережеві сховища чи графічні засоби. На практиці веб-браузер або інша система виконання забезпечує API для вводу-виводу.

Спочатку двигуни JavaScript використовувались лише у веб-браузерах, але зараз вони є основними компонентами інших програмних систем, зокрема серверів та різноманітних додатків.

Амбітна робота над мовою продовжувалась протягом декількох років, завершившись великою колекцією доповнень та уточнень, яка була оформлена публікацією ECMAScript 6 у 2015 році.

Наразі проект специфікації відкрито зберігається на GitHub, а видання ECMAScript видаються за допомогою регулярних щорічних знімків. Потенційний перегляд мови перевіряється шляхом всебічного процесу подання пропозицій. Тепер замість номерів видань розробники перевіряють статус майбутніх функцій індивідуально.

Поточна екосистема JavaScript має безліч бібліотек та фреймворків, усталену практику програмування та розширене використання JavaScript поза веб-браузерами. Плюс, із зростанням односторінкових додатків та інших веб-сайтів, важких для JavaScript, було створено ряд транспіляторів для сприяння процесу розробки.

Використання JavaScript розширилось за межі своїх веб-браузерів. Двигуни JavaScript тепер вбудовані в безліч інших програмних систем, як для розгортання веб-сайтів на стороні сервера, так і для не браузерних програм.

Початкові спроби сприяти використанню JavaScript на стороні сервера були Netscape Enterprise Server та Інтернет-інформаційні служби Microsoft, але вони були невеликими нішами. Використання на стороні сервера врешті-решт почало зростати в кінці 2000-х років, із створенням Node.js та інших підходів.

Electron, Cordova, React Native та інші фреймворки додатків були використані для створення багатьох додатків з поведінкою, реалізованою в JavaScript. Інші програми, що не належать до браузера, включають підтримку Adobe Acrobat для створення сценаріїв документів PDF та розширення оболонки GNOME, написані на JavaScript.

JavaScript нещодавно почав з'являтися в деяких вбудованих системах, як правило, за допомогою Node.js.

JavaScript і DOM дають можливість зловмисним авторам доставляти сценарії для запуску на клієнтському комп'ютері через Інтернет. Автори браузера мінімізують цей ризик, використовуючи два обмеження. По-перше, сценарії працюють у пісочниці, в якій вони можуть виконувати лише дії, пов'язані з Інтернетом, а не загальні завдання програмування, такі як створення файлів.

По-друге, сценарії обмежуються політикою того самого походження: скрипти з одного веб-сайту не мають доступу до такої інформації, як імена користувачів, паролі чи файли cookie, що надсилаються на інший сайт. Більшість помилок безпеки, пов’язаних з JavaScript, є порушенням тієї самої політики походження або пісочниці.

Є підмножини загальних JavaScript - ADsafe, Secure ECMAScript (SES) - які забезпечують більший рівень безпеки, особливо для коду, створеного третіми сторонами (наприклад, реклами). Caja - ще один проект для безпечного вбудовування та ізоляції сторонніх JavaScript та HTML.

Політика безпеки вмісту - основний передбачуваний метод забезпечення того, щоб на веб-сторінці виконувався лише надійний код.

Поширеною проблемою безпеки, пов'язаної з JavaScript, є міжсайтовий сценарій (XSS), порушення політики того самого походження. Уразливості XSS виникають, коли зловмисник може змусити цільовий веб-сайт, наприклад веб-сайт Інтернет-банкінгу, включити шкідливий сценарій у веб-сторінку, представлену жертві. Потім сценарій у цьому прикладі може отримати доступ до банківської програми з привілеями жертви, потенційно розкриваючи секретну інформацію або переказуючи гроші без дозволу жертви. Рішенням вразливостей XSS є використання екрануючого HTML при відображенні ненадійних даних.

Деякі браузери включають частковий захист від відображених атак XSS, в якому зловмисник надає URL-адресу, включаючи шкідливий сценарій. Однак навіть користувачі цих браузерів вразливі до інших атак XSS, таких як ті, де шкідливий код зберігається в базі даних. Тільки правильний дизайн веб-додатків на стороні сервера може повністю запобігти XSS.

Уразливості XSS також можуть виникати через помилки впровадження авторами браузера.

Ще однією уразливістю між сайтами є підробка між запитами (CSRF). У CSRF код на сайті зловмисника змушує браузер жертви вчинити дії, які користувач не передбачав на цільовому сайті (наприклад, переказувати гроші в банку). Коли цільові сайти покладаються виключно на файли cookie для автентифікації запитів,

запити, що походять із коду на сайті зловмисника, можуть мати однакові дійсні дані для входу користувача-ініціатора. Загалом, рішення CSRF полягає у тому, щоб вимагати значення автентифікації у прихованому полі форми, а не лише у файлах cookie, для автентифікації будь-якого запиту, який може мати тривалі наслідки. Також може допомогти перевірка заголовка HTTP Referrer.

"Викрадення JavaScript" - це тип атаки CSRF, при якому тег <script> на сайті зловмисника експлуатує сторінку на сайті жертви, яка повертає приватну інформацію, таку як JSON або JavaScript. Можливі рішення включають вимагання маркера автентифікації в параметрах POST і GET для будь-якої відповіді, яка повертає приватну інформацію.

Розробники клієнт-серверних програм повинні усвідомити, що ненадійні клієнти можуть перебувати під контролем зловмисників. Автор програми не може припустити, що їх код JavaScript працюватиме належним чином (або взагалі), оскільки будь-яка таємниця, вбудована в код, може бути вилучена визначеним супротивником. Деякі наслідки:

Автори веб-сайтів не можуть ідеально приховати, як працює їх JavaScript, оскільки вихідний код повинен надсилатися клієнту. Код можна заплутати, але затухання можна змінити.

Перевірка форми JavaScript забезпечує лише зручність для користувачів, а не безпеку. Якщо сайт підтверджує, що користувач погодився з його умовами обслуговування, або фільтрує недійсні символи з полів, які повинні містити лише цифри, він повинен це робити на сервері, а не лише на клієнті.

Сценарії можна вибірково відключити, тому на JavaScript не можна покладатися, щоб запобігти таким операціям, як клацання правою кнопкою миші на зображенні, щоб зберегти його.

Вбудовувати в JavaScript конфіденційну інформацію, таку як паролі, вважається дуже поганою практикою, оскільки її може отримати зловмисник.

Системи управління пакетами, такі як npm та Bower, популярні серед розробників JavaScript. Такі системи дозволяють розробнику легко управляти залежностями своєї програми від інших бібліотек програм розробника. Розробники

вірять, що працівники бібліотек забезпечуватимуть їх безпеку та оновлення, але це не завжди так. Через цю сліпу довіру виникла вразливість. Бібліотеки, на які покладаються, можуть мати нові випуски, які спричиняють появу помилок або уразливостей у всіх програмах, які покладаються на бібліотеки. І навпаки, бібліотека може не виправлятися з відомими вразливими місцями в дикій природі. В ході дослідження, проведеного на вибірці з 133 тис. Веб-сайтів, дослідники виявили, що 37% веб-сайтів включали бібліотеку з принаймні однією відомою вразливістю. "Середнє відставання між найстарішою версією бібліотеки, що використовується на кожному веб-сайті, та найновішою доступною версією цієї бібліотеки становить 1177 днів у ALEXA, а розвиток деяких бібліотек, які все ще активно використовуються, припинився роками тому". Інша можливість полягає в тому, що супровідник бібліотеки може повністю видалити її. Це сталося в березні 2016 року, коли Азер Кочулу видалив своє сховище з npm. Це призвело до поломки всіх десятків тисяч програм та веб-сайтів, залежно від його бібліотек.

JavaScript надає інтерфейс для широкого спектру можливостей браузера, деякі з яких можуть мати такі недоліки, як переповнення буфера. Ці недоліки можуть дозволити зловмисникам писати сценарії, які запускатимуть будь-який бажаний код в системі користувача. Цей код жодним чином не обмежується іншим додатком JavaScript. Наприклад, експлойт перевищення буфера може дозволити зловмиснику отримати доступ до API операційної системи з привілеями суперкористувача.

Ці недоліки вплинули на основні браузери, включаючи Firefox, Internet Explorer і Safari.

Плагіни, такі як відеоплеєри, Adobe Flash та широкий спектр елементів керування ActiveX, увімкнених за замовчуванням у Microsoft Internet Explorer, також можуть мати недоліки, які можна використати за допомогою JavaScript (такі недоліки використовувались у минулому).

У Windows Vista Microsoft намагалася стримувати ризики помилок, таких як переповнення буфера, запускаючи процес Internet Explorer з обмеженими

привілеями. Google Chrome аналогічним чином обмежує свої візуатори сторінок власною "пісочницею".

Веб-браузери здатні запускати JavaScript поза пісочницею, маючи привілеї, необхідні, наприклад, для створення або видалення файлів. Такі привілеї не призначені для надання коду з Інтернету.

Неправильне надання привілеїв JavaScript з Інтернету зіграло свою роль у вразливостях як в Internet Explorer, так і у Firefox. У Windows XP з пакетом оновлень 2 Microsoft знизила привілеї JScript в Internet Explorer.

Microsoft Windows дозволяє запускати вихідні файли JavaScript на жорсткому диску комп’ютера як загальні програми, що не містять середовища (див .: Windows Script Host). Це робить JavaScript (як і VBScript) теоретично життєздатним вектором для троянського коня, хоча троянські коні JavaScript на практиці трапляються рідко.

2.2 React

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript із відкритим кодом, інтерфейс, для створення інтерфейсів користувача або компонентів інтерфейсу. Він підтримується Facebook та спільнотою окремих розробників та компаній. React можна використовувати як основу при розробці односторінкових або мобільних додатків. Однак React займається лише управлінням станом та наданням цього стану в DOM, тому для створення програм React зазвичай потрібно використовувати додаткові бібліотеки для маршрутизації, а також певну функціональність на стороні клієнта.

Далі наведено елементарний приклад використання React у HTML із JSX та JavaScript.

```
<div id="myReactApp"></div>
<script type="text/babel">
  function Greeter(props) {
```

```
return <h1>{props.greeting}</h1>
  }
  var App = <Greeter greeting="Hello World!" />;
  ReactDOM.render(App, document.getElementById('myReactApp'));
</script>
```

Функція Greeter - це компонент React, який приймає привітання властивості. Змінна App є екземпляром компонента Greeter, де для властивості привітання встановлено значення «Hello World!». Потім метод ReactDOM.render відображає наш компонент Greeter всередині елемента DOM з ідентифікатором myReactApp.

При відображенні у веб-браузері результат буде

```
<div id = "myReactApp">
  <h1> Hello World! </h1>
</div>
```

Код реагування складається з сутностей, які називаються компонентами. Компоненти можуть бути відтворені до певного елемента в DOM за допомогою бібліотеки React DOM. При візуалізації компонента можна передавати значення, які відомі як "реквізит":

```
ReactDOM.render(<Greeter greeting="Hello World!" />, document.getElementById('myReactApp'));
```

Ще однією помітною особливістю є використання віртуальної об'єктної моделі документа або віртуальної DOM. React створює кеш-структуру даних в пам’яті, обчислює отримані різниці, а потім ефективно оновлює відображуваний DOM браузера. Цей процес називається примиренням. Це дозволяє програмісту писати код так, ніби вся сторінка відображається при кожній зміні, тоді як бібліотеки React відображають лише підкомпоненти, які насправді змінюються. Цей вибірковий рендеринг забезпечує значне підвищення продуктивності. [Потрібне цитування] Це економить зусилля з перерахунку стилю CSS, макета для сторінки та рендерингу для всієї сторінки.

Методи життєвого циклу використовують форму підключення, яка дозволяє виконувати код у задані точки протягом життя компонента.

- shouldComponentUpdate дозволяє розробнику запобігати непотрібному повторному рендерингу компонента, повертаючи значення false, якщо візуалізація не потрібна.
- componentDidMount викликається, як тільки компонент "змонтований" (компонент створюється в інтерфейсі користувача, часто шляхом асоціювання його з вузлом DOM). Це зазвичай використовується для запуску завантаження даних з віддаленого джерела через API.
- componentWillUnmount викликається безпосередньо перед тим, як компонент буде зруйнований або "демонтований". Це зазвичай використовується для очищення залежностей від компонента, що вимагають ресурсів, і які не будуть просто видалені при демонтажі компонента (наприклад, видалення будь-яких екземплярів setInterval (), які пов'язані з компонентом, або "eventListener", встановленого на " документ "через наявність компонента)
- візуалізація - це найважливіший метод життєвого циклу і єдиний необхідний у будь-якому компоненті. Зазвичай він викликається кожного разу, коли стан компонента оновлюється, що повинно відображатися в інтерфейсі користувача.

Для підтримки концепції React про односпрямований потік даних (який може бути протиставлений двонаправленому потоку AngularJS), архітектура Flux була розроблена як альтернатива популярній архітектурі модель-вигляд-контролер. Flux має дії, які надсилаються через центральний диспетчер до магазину, а зміни в магазині передаються назад до подання. При використанні з React це розповсюдження здійснюється завдяки властивостям компонентів. З моменту своєї концепції Flux був замінений такими бібліотеками, як Redux та MobX.

Потік можна вважати варіантом моделі спостерігача.

Компонент React під архітектурою Flux не повинен безпосередньо модифікувати будь-який переданий йому реквізит, але повинен передавати функції зворотного виклику, які створюють дії, які відправляються диспетчером для модифікації сховища. Дія - це об'єкт, відповідальність якого полягає в описі того,

що відбулося: наприклад, дія, що описує одного користувача, "що слідує" за іншим, може містити ідентифікатор користувача, цільовий ідентифікатор користувача та тип USER_FOLLOWED_ANOTHER_USER. Магазини, які можна сприймати як моделі, можуть змінити себе у відповідь на дії, отримані від диспетчера.

Ця закономірність іноді виражається як "властивості стікають вниз, дії течуть вгору". З моменту його створення було створено багато реалізацій Flux, мабуть, найвідомішим є Redux, який має єдиний магазин, який часто називають єдиним джерелом істини.

React був створений Джорданом Уолком, інженером-програмістом у Facebook, який випустив ранній прототип React під назвою "FaxJS". На нього вплинув XHP, бібліотека HTML-компонентів для PHP. Вперше він був розміщений на стрічці новин Facebook у 2011 році, а пізніше в Instagram у 2012 році. Він був відкритий у JSConf US у травні 2013 р.

React Native, який дозволяє розробляти власні Android, iOS та UWP разом з React, був анонсований на React Conf у Facebook у лютому 2015 року та відкритим у березні 2015 року.

18 квітня 2017 року Facebook оголосив React Fiber, новий базовий алгоритм бібліотеки React для побудови користувальницьких інтерфейсів. React Fiber повинен був стати основою будь-яких майбутніх удосконалень та розвитку функцій бібліотеки React.

26 вересня 2017 року React 16.0 був випущений для широкого загалу.

16 лютого 2019 року React 16.8 був оприлюднений для громадськості. Випуск представив React Hooks.

10 серпня 2020 року команда React оголосила першого кандидата на випуск React v17.0, який відзначається першим серйозним випуском без серйозних змін в API-інтерфейсі React для розробників.

2.3 React Native

React Native - це фреймворк JavaScript, який використовується для написання власних рендеринг мобільних додатків з перехресними функціоналами. Це означає, що програми, розроблені React Native, можна використовувати як на iOS, так і на Android.

React Native базується на JavaScript-бібліотеці від Facebook під назвою React. Вона була створена для побудови користувальницьких інтерфейсів для браузерів, а пізніше на Facebook створено мобільний фреймворк під назвою React Native. Веб-розробникам, які мають досвід роботи з JavaScript та React - дозволяє легко писати, знайомою мовою, додатки для всіх мобільних пристроїв одночасно.

Поведінка React Native дуже схожа на програми React оскільки вони обидва використовують комбінацію розмітки JavaScript і XML, відому як JSX. React Native викликає власне відображення API в Objective-C для iOS та Java для Android. Таким чином мобільні додатки відображаються за допомогою реальних компонентів мобільного інтерфейсу користувача і тому виглядають як будь-які інші мобільні додатки.

2.3.1 Переваги React Native

Однією з найбільших переваг є те, що React Native за допомогою платформи розміщує стандартні API. Цей власний прийом візуалізації не використовується іншими крос-платформними фреймворками. Інші фреймворки, прагнуть досягти чогось подібного за допомогою комбінації JavaScript, HTML і CSS і відображають їх за допомогою Web Views, що являє собою спрощений веб-браузер всередині мобільного додатка. Тому він може відображати елементи власного інтерфейсу користувача на Android, iOS та Windows Phone.

Продуктивність важлива для мобільних додатків і React Native може досягти високої продуктивності за допомогою того, що працює окремо від основного

потоку інтерфейсу користувача. React Native візуалізує перегляди шляхом зміни реквізиту або стану. Реквізит - це так звані властивості що кожен компонент містить. Додаток React Native складається з компонентів і кожен компонент представляє частину графічного інтерфейсу, який відображається на мобільному телефоні. Він також може включати деякі функції виконання операцій в мобільному додатку. Це ефективний спосіб спілкування між компонентами, які відображають інтерфейс користувача.

React Native та React змінюють та модифікують вміст у графічному інтерфейсі за допомогою зміни стану та реквізиту. React Native змінює стан і реквізити шляхом мутації бібліотек користувальницького інтерфейсу, наданих мобільним телефоном. Це робиться мобільними виробниками, щоб гарантувати, що відображається правильний вміст та розроблений мобільний додаток не створює проблем із продуктивністю. React змінює стан і реквізит за допомогою розмітки HTML і CSS.

Додаткова перевага середовища розробки в React Native те, що зміни в мобільному додатку можуть бути миттєві із попереднім переглядом. Існує також потужний інструмент налагодження, інтегрований в React Native, який використовує браузер для презентації інформації налагодження. Під час розробки розробник може налагодити мобільний додаток одночасно із поданням даних налагодження у веб-браузері.

Повторне використання коду з React Native дуже зручне. React Native додатки побудовані з компонентів, які можна повторно використовувати.

Весь код не можна використовувати повторно, оскільки він націлений на різні платформи.

Facebook і Microsoft працюють над багаторазовим використанням, оскільки воно може заощадити багато часу під час розробки.

2.4 Redux

React Native підходить для написання невеликих додатків, але коли додаток починає рости стає все більш складною обробка всього штату і реквізиту, якщо багато компонентів. Тому була розроблена бібліотека під назвою Redux для вирішення цієї складності.

У наш час користувальницькі інтерфейси стають все більш складними та потребують досить багато обслуговування. Маршрутизація часто реалізується на клієнті так що нам не потрібно оновлювати браузер, щоб побачити деякі зміни. До цього програма мала оновити всю сторінку. Маршрутизація на стороні клієнта вигідна для продуктивності, але це означає що клієнт повинен обробляти більше станів порівняно з сервером. Управління всіма цими станами може бути дуже складним, якщо не вирішити проблему правильно.

Actions є об'єктами JavaScript, які описують зміни стану в додатку для різних компонентів. Тоді йдуть action creators, які в основному поводяться як функції, які приймають параметри і повертають action там, де вони повинні щось змінити. Редуктори, які отримують дії та поточний стан, повертають новий стан і надсилають його в store. Store поводиться як ядро Redux і зберігає та охороняє state додатку. Всі компоненти в рамках програми можуть підписатися на зміни state в store та відправляти actions до нього. Завдяки цьому компоненти можуть спілкуватися між собою, а не переходити до батьківського вузла кожного разу.

3 РОЗРОБКА ДОДАТКА

3.1 Вирішення проблем - React Native

Розробка на React Native створила деякі проблеми через відсутність у автора знань про JavaScript та React. Отже, спочатку були певні труднощі, але з наполегливістю, пошуком інформації та швидкою перевіркою нестачі знань, вони були перетворені на ефективний процес розвитку через приклади коду, дослідження та тестування. Це починало приносити більше задоволення і переваги React Native стали яснішими.

Одна з перших проблем що траплялася під час розробки, була обробка запитів між компонентами в додатку. Компонент відображає частину графічного інтерфейсу і зв'язок між компонентами може швидко стати дуже складним, якщо структура неефективна.

Проблема введення Redux у вже розроблену систему може зайняти багато часу, однак це може сприяти створенню ефективної системи.

3.2 Розробка

Для того, щоб встановити React Native, потрібно встановити Node.js. За допомогою менеджера пакетів вузлів (npm), який є доповненням до Node, розробник може встановити React Native за допомогою командного рядка. Для того, щоб запустити додаток React Native в Android, на комп'ютері потрібно встановити Java JDK та Android SDK разом з Android SDK Build-Tools 23.0.1. Після встановлення React Native можна створити та запустити нову програму React Native, ще раз ввівши нову команду.

Щойно створений проект Project містить:

• package.json - файл, що містить відповідні метадані проекту або залежності проекту.

• node_modules / - Містить залежності та інструмент CLI, node_modules / responsenative / local-cli / cli.js, який керує проектом. Цей сценарій виконує інший допоміжний сценарій node_modules / response-native / init.sh, який збирає шаблонний код.

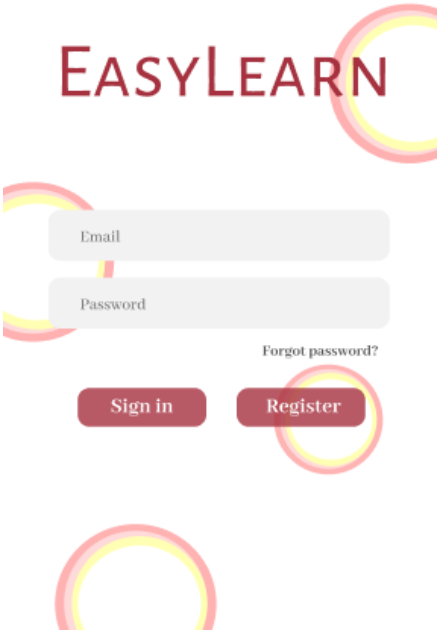
• index.js - головний файл проекту React Native.

Як уже зазначалося, файл index.js є основним файлом коду і містить основні функції з самого початку. Перш за все, перелічено імпорт. Програма перелічує необхідний імпорт з модулів, тобто компоненти, таблиці стилів, зображення тощо. Цей імпорт потім може бути використаний для створення компонентів, які можуть бути відтворені в додатку.

Як вже згадувалося раніше, додаток React Native створюється за допомогою компонентів React у формі об'єктів JavaScript, які відповідають за візуалізацію та автоматично оновлюють інтерфейс користувача. Перший компонент, який створюється, - це компонент нестандартного класу, компонент, який є першим екраном програми. Цей компонент включає функцію візуалізації, що містить макет, написаний у JSX.

3.3 Easy Learn - React Native

Перший прототип мобільного додатка, було розроблено на React Native. Однією з перших речей, яка була реалізована, є система входу, що зображена на малюнку 3.1.



Малюнок 3.1 - Тут користувач авторизується за допомогою електронної пошти та пароля.

При спробі ввійти в систему програма також обробляє всі запити з показниками активності з метою створення візуального зворотного зв'язку з користувачем. Це є дуже важливо при взаємодії з мобільним додатком, тому що немає іншого візуального зворотного зв'язку, який надсилається користувачеві і тому може бути незрозуміло, як просуваються різні види діяльності.

Якщо користувач не ввійшов до системи, поля введення електронної пошти і пароля стають порожніми, і користувач може спробувати ввійти ще раз.

Користувачеві дозволяється необмежена кількість спроб увійти в систему. Якщо користувач успішно увійшов, користувач буде направлений на головну сторінку, див. малюнок 3.2.

Малюнок 3.2 - Головна сторінка.

На головній сторінці представлені всі курси, кнопка пошуку та 3 кнопки навігації, див. малюнок 3.2.

Перша кнопка у вигляді домівки - кнопка з посиланням на головну сторінку. Друга – це сторінка де всі курси розбиті по категоріях. Третя кнопка використовується для відкривання сторінки з обраними курсами, які можна додавати за допомогою іконки в формі серця, вона знаходиться під описом курсу, див. малюнок 3.3.

Малюнок 3.3 – Сторінка з обраними курсами.

Також вгорі знаходиться кнопка в формі лупи, при натисканні на яку з’являється поле для вводу назви курсу, або ключових слів для пошуку, див.малюнок 3.4.

Малюнок 3.4 – Реалізація пошуку.

Головна сторінка

На головній сторінці реалізовано представлення усіх курсів з таких порталів як Udemy та Prometheus, вони оновлюються в режимі реального часу, тобто коли на один з цих сайтів додається новий курс, він одразу ж з’являється в додатку. Дана функція розроблена з використанням API цих порталів, детальніше вони описані в наступному розділі. Ми посилаємо GET запит на сервер, наприклад, Udemy і у відповідь отримуємо JSON, який потрібно розпарсити і вивести потрібні нам дані на екран. Потрібно зазначити, що ці дані ми не зберігаємо і опрацьовуємо асинхронно, тобто, при поганому з’єднанні користувач може побачити не всю інформацію одразу, або не побачити, наприклад, картинку до курсу, але вона з’явиться з часом.

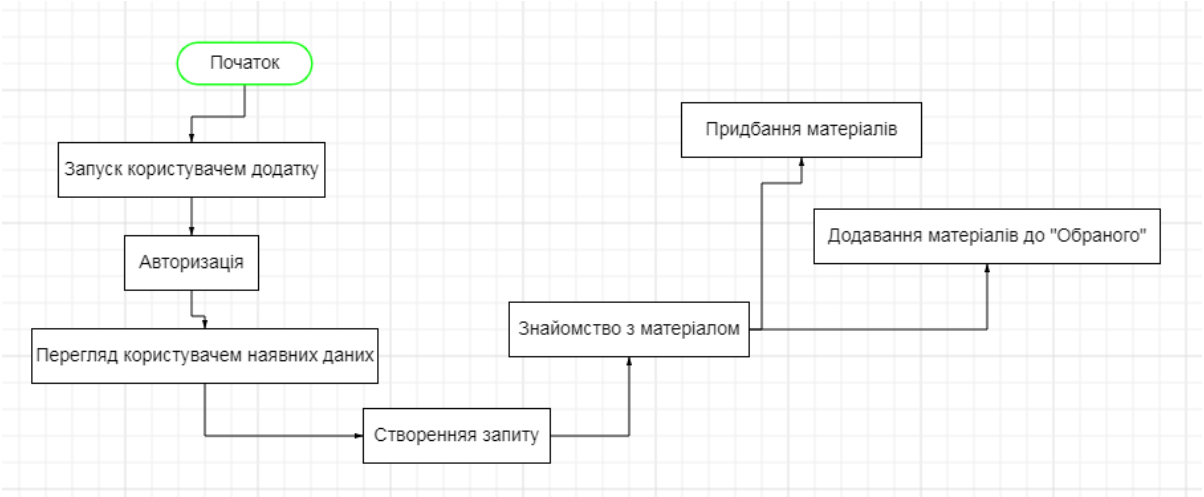
Як представлені курси

Курси представлені у вигляді блоків, які складаються з картинки, назви, короткого опису, та ціни за цей курс, див.малюнок 3.5.

Малюнок 3.5 - Опис обраного матеріалу

Якщо користувач хоче відкрити цей курс і прочитати інформацію повністю, або придбати його, то така можливість є, при натисканні кнопку «Buy Now» поруч із ціною. В такому випадку в браузері відкриється сайт, на якому знаходиться цей курс - Udemy.com або Prometheus.io.

Діаграма діяльності додатка – малюнок 3.6.



Малюнок 3.6- Діаграма діяльності додатка

Сторінка «Вибрані»

На цій сторінці представлені курси, які користувач відзначив іконкою «серце», для того, щоб не загубити, або порівняти з іншими. Також курси можна відсортувати за ціною, датою, або популярністю, і в будь-який момент видалити з цього списку.

Сторінка «Категорії»

На цій сторінці можна обрати категорію яка цікавить. На основі інтересів користувача будуть з’являтись курси на головній сторінці. Це було реалізовано завдяки API, в якому є можливість сортування за інтересами.

3.4 API’s

Абревіатура API розшифровується як «Application Programming Interface» (інтерфейс програмування додатків, програмний інтерфейс програми). Більшість великих компаній на певному етапі розробляють API для клієнтів або для внутрішнього використання. Щоб зрозуміти, як і яким чином API застосовується в розробці і бізнесі, спочатку потрібно розібратися, як влаштована «всесвітня павутина».

Всесвітня павутина й віддалені сервери

WWW можна уявити як величезну мережу пов'язаних серверів, на яких і зберігається кожна сторінка. Звичайний ноутбук можна перетворити в сервер, здатний обслуговувати цілий сайт в мережі, а локальні сервери розробники використовують для створення сайтів перед тим, як відкрити їх для широкого кола користувачів. При введенні в адресний рядок браузера www.facebook.com на віддалений сервер Facebook відправляється відповідний запит. Як тільки браузер отримує відповідь, то інтерпретує код і відображає сторінку.

Кожен раз, коли користувач відвідує якусь сторінку в мережі, він взаємодіє з API віддаленого сервера. API - це складова частина сервера, яка отримує запити і відправляє відповіді.

API як спосіб обслуговування клієнтів

Багато компаній пропонують API як готовий продукт. Наприклад, Weather Underground продає доступ до свого API для отримання метеорологічних даних.

Сценарій використання: на сайті невеликої компанії є форма для запису клієнтів на прийом. Компанія хоче вмонтувати в нього Google Календар, щоб дати клієнтам можливість автоматично створювати події і вносити деталі про майбутній зустрічі.

Застосування API: мета - сервер сайту повинен безпосередньо звертатися до сервера Google із запитом на створення події з зазначеними деталями, отримувати

відповідь Google, обробляти її, і передавати відповідну інформацію в браузер, наприклад, повідомлення із запитом на підтвердження користувачеві.

В якості альтернативи браузер може зробити запит до API сервера Google, минаючи сервер компанії.

Чим API Google Календаря відрізняється від API будь-якого іншого віддаленого сервера в мережі?

Технічно, різниця в форматі запиту і відповіді. Щоб згенерувати повну веб-сторінку, браузер очікує відповідь на мові розмітки HTML, в той час як API Google Календаря поверне просто дані в форматі JSON.

Якщо запит до API робить сервер веб-сайту компанії, то він і є клієнтом (так само, як клієнтом виступає браузер, коли користувач відкриває веб-сайт).

Користувач завдяки API отримує можливість вчинити дію, не залишаючи сайт компанії.

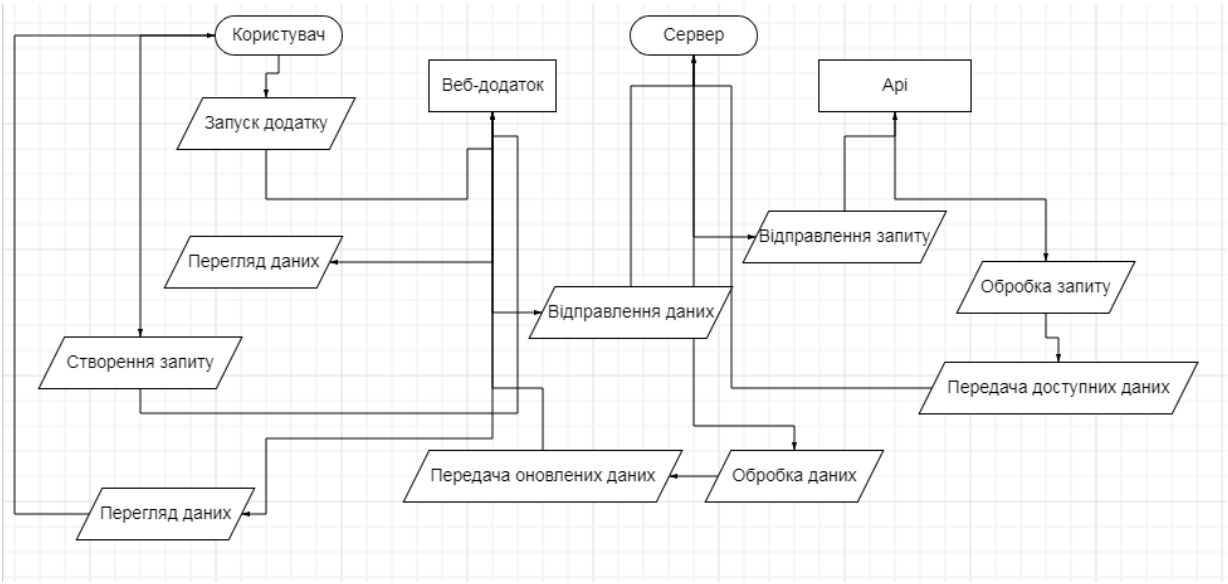
Більшість сучасних сайтів використовують принаймні кілька сторонніх API. Багато задач вже мають готові рішення, пропоновані сторонніми розробниками, будь то бібліотека або послуга. Найчастіше простіше і надійніше вдатися саме до вже готового рішення.

Часто розробники розносять додаток на кілька серверів, які взаємодіють між собою за допомогою API. Сервери, які виконують допоміжну функцію по відношенню до головного сервера додатку, називаються мікросервісами.

Таким чином, коли компанія пропонує своїм користувачам API, це просто означає, що вона створила ряд спеціальних URL, які в якості відповіді повертають тільки дані.

Такі запити часто можна відправляти через браузер. Так як передача даних по протоколу HTTP відбувається в текстовому вигляді, браузер завжди зможе відобразити відповідь. Наприклад, через браузер можна безпосередньо звернутися до API GitHub (https://api.github.com/users/example), причому без маркера доступу.

Блок-схема принципу роботи даного додатку- малюнок 3.7.



Малюнок 3.7 – Діаграма принципу роботи додатка.

Ще кілька прикладів API

Слово «application» (прикладний, додаток) може застосовуватися в різних значеннях. В контексті API воно має на увазі:

- фрагмент програмного забезпечення з певною функцією сервер повністю, додаток повністю або ж просто окрему частину програми

В об'єктно-орієнтованому проектуванні код представлений у вигляді сукупності об'єктів. У додатку таких об'єктів, що взаємодіють між собою, можуть бути сотні. У кожного з них є свій API - набір публічних властивостей і методів для взаємодії з іншими об'єктами в додатку. Об'єкти можуть також мати приватну, внутрішню логіку, яка прихована від оточення і не є API.

3.4.1 Udemy API

Udemy API для партнерів дає розробникам широкий доступ до функціональності Udemy для розробки клієнтських додатків і інтеграції Udemy.

Все організовано навколо REST. API розроблений таким чином, щоб мати передбачувані, орієнтовані на ресурси URL і використовувати коди відповідей HTTP для вказівки помилок API. Він використовує вбудовані функції HTTP, наприклад, автентифікацію HTTP і команди HTTP, які розуміють готові клієнти HTTP. Він приймає тільки виклики https до API. Всі відповіді будуть повернуті в форматі JSON, включно з помилками. Протокол OAuth 2 використовується для авторизації користувачів (ще не задокументовано).

Поточна версія Udemy API для партнерів - 2.0, коренева кінцева точка для всіх ресурсів - https://www.udemy.com/api-2.0/.

Для виконання будь-яких викликів до Udemy REST API необхідно створити API-клієнт. API-клієнт для партнерів складається з токена на пред'явника (bearer token), який підключений до аккаунту користувача Udemy.

Якщо ви хочете створити API-клієнт для партнера, зареєструйтесь на udemy.com і перейдіть в розділ API-клієнти вашого профілю користувача. Після схвалення запиту ваш щойно створений API-клієнт для партнера буде активований, а ваш токен на пред'явника (bearer token) почне відображатися на сторінці API-клієнти. Для перевірки клієнта Udemy API для партнерів вимагає введення основних параметрів автентифікації. Параметри автентифікації повинні відправлятися при кожному виклику. В іншому випадку буде відображатися помилка 401 UNAUTHORIZED.

Для відправки запитів на перевірку користувача вкажіть значення client_id і client_secret як заголовок авторизації HTTP Authorization в кодуванні base64.

curl --user {YOUR_CLIENT_ID}: {YOUR_CLIENT_SECRET} https://www.udemy.com/api-2.0/courses/

curl -H "Authorization: Basic {BASE64_ENCODED (CLIENT_ID: CLIENT_SECRET)}" https://www.udemy.com/api-2.0/courses/

Udemy API для партнерів підтримує додатковий параметр запиту fields для налаштування полів, які повинні бути повернуті для конкретного типу об'єкта. Параметр fields є хеш-картою, яка зазвичай визначається як параметри GET типу

fields [course], fields [user] і т. д. Кожен тип об'єкта також має списки полів @min, @default і @all.

Приклади

Повернути об'єкт курсу з полями назви і заголовка:

https://www.udemy.com/api-2.0/courses/238934/?fields[course]=title,headline

Повернути об'єкт курсу з усіма полями в @min і власника, але за винятком зображень. Для об'єкта користувача повертається тільки поле заголовка:

https://www.udemy.com/api-2.0/courses/238934/?fields[course]=@min,owner,-images&fields[user]=title

Більшість API видають відповіді з розбивкою на сторінки. Розмір сторінки можна змінити, вказавши в параметрі запиту page_size. Розмір сторінки за замовчуванням становить 12. Максимальний розмір сторінки становить 100. Сторінку можна змінити, вказавши параметр запиту page.

Udemy API використовують конвенціальні коди запитів HTTP, щоб показати успіх або відмову запиту API. У загальному випадку, коди в діапазоні 2xx показують успіх, коди в діапазоні 4xx показують помилку, яка виникла через надану інформацію (наприклад, відсутній необхідний параметр, не вдалося оплатити і т.д.), коди в діапазоні 5xx показують помилку на серверах Udemy.

3.4.2 Prometheus HTTP API

Поточний стабільний HTTP API доступний за адресою /api/v1 на сервері Prometheus. Будь-які доповнення будуть додані до цієї кінцевої точки.

Формат відповіді API - JSON. Кожен успішний запит API повертає код стану 2xx. Недійсні запити, які надходять до обробників API, повертають об’єкт помилки в форматі JSON та один із кодів відповіді HTTP:

- 400 Помилковий запит, якщо параметри відсутні або неправильні.

- 422 Неопрацьована сутність, коли вираз неможливо виконати (RFC4918).
- 503 Послуга недоступна, коли термін очікування або скасування запитів.

Інші коди, що не є 2xx, можуть бути повернуті за помилки, що виникають до досягнення кінцевої точки API.

Масив попереджень може бути повернутий, якщо є помилки, які не перешкоджають виконанню запиту. Усі дані, які були успішно зібрані, будуть повернуті в поле даних.

Загальні заповнювачі визначаються таким чином:

- <rfc3339 | unix_timestamp>: Вхідні мітки часу можуть бути надані або у форматі RFC3339, або як мітка часу Unix у секундах, з необов'язковими десятковими знаками для точності до секунди. Вихідні мітки часу завжди представляються як мітки часу Unix у секундах.
- <series_selector>: селектори часових рядів Prometheus, такі як http_requests_total або http_requests_total {method = ~ "(GET | POST)"}, і їх потрібно закодувати за URL-адресою.
- <duration >: рядки тривалості Prometheus. Наприклад, 5m відноситься до тривалості 5 хвилин.
- <bool>: булеві значення (рядки true і false).

Примітка: Назви параметрів запиту, які можна повторити, закінчуються на [].

ВИСНОВКИ

У процесі виконання дипломної роботи було здійснено:

- Пошук існуючих аналогів та їх аналіз.
- Модель об’єкту проектування
- Розробка функціоналу мобільного додатку
- Перевірка функціональності

Для подальшого розвитку додатку необхідно розширювати його функціональність. Наприклад:

- Додати інші популярні джерела освітньої інформації.
- Розробити API, для можливості подальшого використання зібраної інформації у власному додатку іншими зацікавленими особами.
- Надати можливість педагогам чи освітнім закладам на реєстрацію та можливість ділитися власними навчальними матеріалами у додатку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Джакоб Фрідман та Ерік Мас'єло, Mastering React Native. Packt Publishing, 2017.
2. Офіційна документація фреймворку Redux: https://redux.js.org/.
3. Ресурс для IT-спеціалістів [Електроний ресурс] – 2018. – Режим доступу: www.habr.com.
4. Роберт Мартін, Чиста архітектура. Мистецтво розробки програмного забезпечення – 2018.
5. Технології створення мобільних додатків. URL: http: // group-global. org/ publication / 63343-tehnologiisozdaniya-mobilnyh-prilozheniy.
6. Bill Phillips and Brian Hardy. Android Programming: The Big Nerd Ranch Guide, 1st edition — Published by Big Nerd Ranch Inc., 2013 — P. 221-312.
7. Bonnie Eisenman «Learning React Native». Apress, – 2018. – 227 с.
8. Bonnie Eisenman. Learning React Native: Building Native Mobile Apps with JavaScript, 2nd edition — Published by O'Reilly Media, 2016 — P. 150-400.
9. Ethan Holmes, Tom Bray. Getting Started with React Native — Published by Packt Publishing Ltd, 2015 — P. 129-172.
10. Expo Docs [Електронний Ресурс] – Режим доступу : https://expo.io/.
11. Frank Zammetti. Practical React Native — Published by Apress, 2018 — P. 96-133.
12. Mateusz Grzesiukiewicz. Hands-On Design Patterns with React Native — Published by Packt Publishing Ltd, 2018 — P. 59-90.
13. Microsoft Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/.
14. Mognoose. elegant mongodb object modeling for node.js [Електронний ресурс] : API Docs. Режим доступу : https://mongoosejs.com/docs/api.html.
15. React Native — A framework for building native apps using React [Електронний ресурс] — Режим доступу до ресурсу: https://facebook.github.io/react-native/.

16. React Native [Електронний ресурс]: Documentation. Режим доступу: https://reactnative.dev/docs/getting-started.
17. React Native Share Docs [Електронний ресурс] – Режим доступу: https://github.com/react-native-community/react-native-share.
18. Smile-Ukraine [Електроний ресурс] – 2018. – Режим доступу: smileukraine.com/ua/mobile-apps/mobile-apps-types.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

РОЗРОБКА ДОДАТКУ 'EASY LEARN' З ВИКОРИСТАННЯМ REACT NATIVE

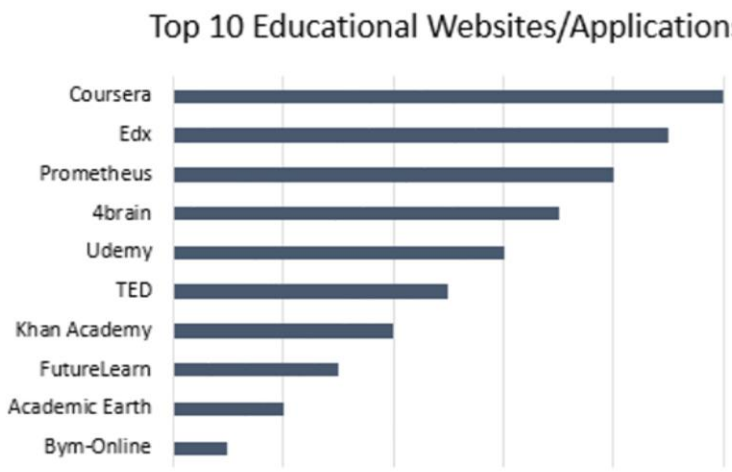
Виконала студентка 4 курсу
групи ПД-44
Рудська Анастасія Ігорівна
Керівник роботи
Гребенюк Віктор Вікторович

Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи-** розробка мобільного додатку для зручності пошуку необхідних навчальних курсів.
- **Об'єкт дослідження-** процес пошуку матеріалів для навчання.
- **Предмет дослідження-** мобільний додаток, для знаходження необхідних навчальних матеріалів.

ПОПУЛЯРНІ ОСВІТНІ ПОРТАЛИ



АНАЛОГИ

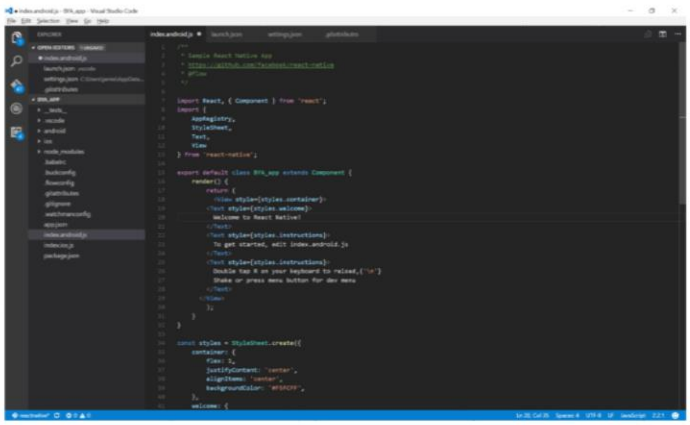
Назва	Переваги	Недоліки
YouTube	Швидкий доступ до відеоматеріалів, можливість створення власного каналу.	Відсутня фільтрація відео на належному рівні, сервіс не працюватиме без доступу до інтернету.
Udemy	Різноманітність курсів, зрозумілий інтерфейс.	Недостовірна система оцінки, частково застарілі матеріали.
Prometheus	Безкоштовні курси.	Необхідність постійного підключення до мережі інтернет.
Coursera	На сайті введено «Кодекс честі», задля уникнення шахрайства.	Надмір інформації на форумах курсу.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

1. Visual Studio 2017 Professional
2. Expo Client

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

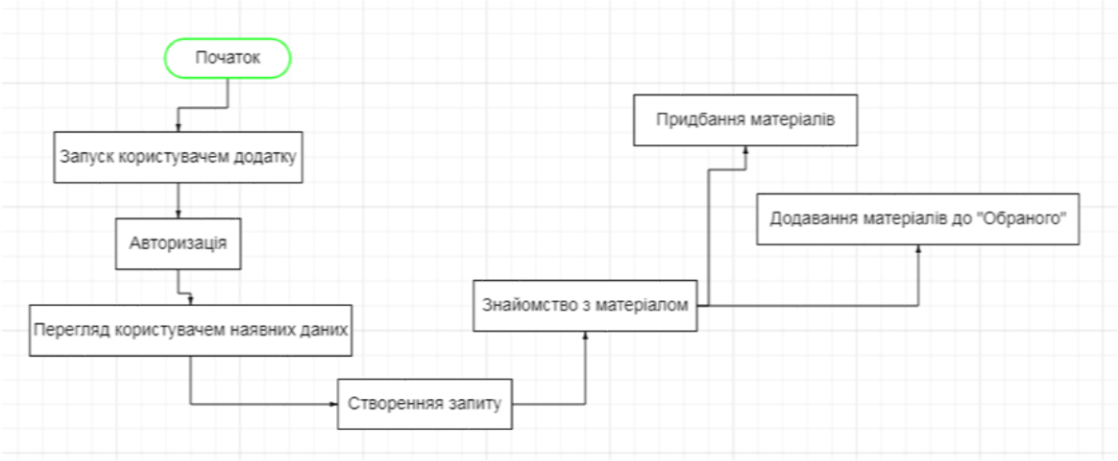
Visual Studio 2017 Professional:



Expo Client:

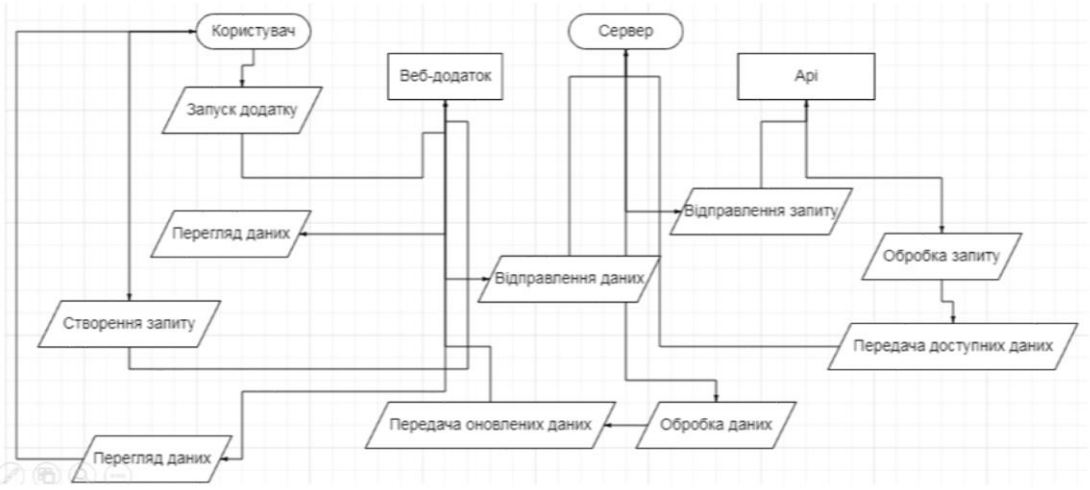
МЕТОДИ ТА КЛАСИ ПРОГРАМИ

- Діаграма діяльності

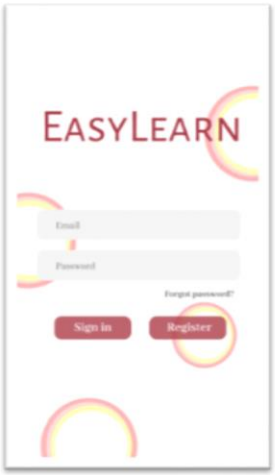


МЕТОДИ ТА КЛАСИ ПРОГРАМИ

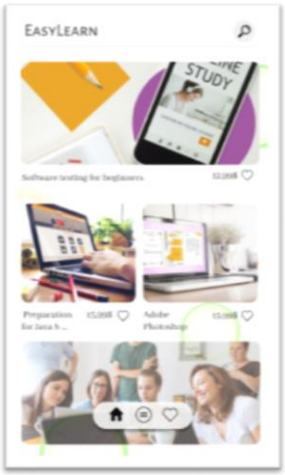
- Діаграма послідовності додатку



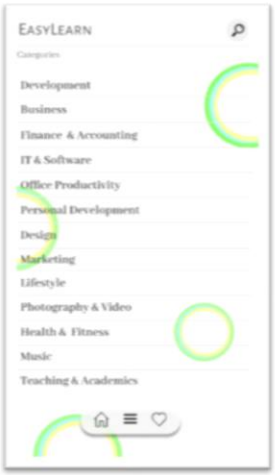
ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА



Авторизація користувача

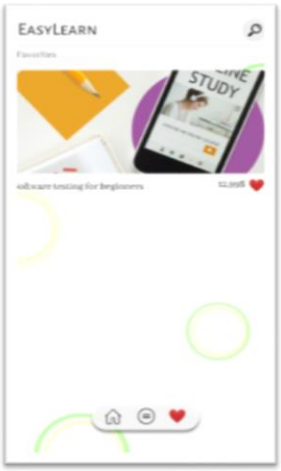


Головна сторінка



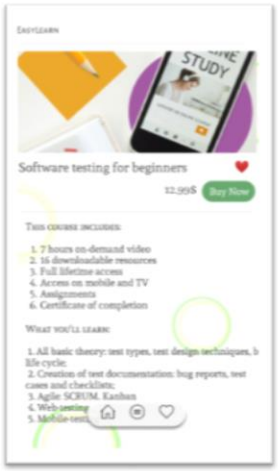
Категорії

ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА



Обрані

Реалізація пошуку



Опис матеріалу

ВИСНОВКИ

У процесі виконання дипломної роботи було здійснено:
1.Пошук існуючих аналогів та їх аналіз.
2.Модель об’єкту проектування
3.Розробка функціоналу мобільного додатку
4.Перевірка функціональності

Для подальшого розвитку додатку необхідно розширювати його функціональність. Наприклад:
1.Додати інші популярні джерела освітньої інформації.
2.Розробити арі, для можливості подальшого використання зібраної інформації у власному додатку іншими зацікавленими особами.
3.Надати можливість педагогам чи освітнім закладам на реєстрацію та можливість ділитися власними навчальними матеріалами у додатку.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

А.І. Рудська, Ковбаса В.В. Автоматизація бізнесу за допомогою ERP-систем / А.І. Рудська, Ковбаса В.В. // Як працювати з анонімними даними у Data Science, на прикладі мобільних компаній : II Всеукраїнська науково-практична конференція «Системний аналіз в бізнесі та управлінні».

ДЯКУЮ ЗА УВАГУ!