Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Веб-клієнт електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua

Виконав: студент 6 курсу, групи СПм-61
спеціальності 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

(підпис) Дрозд О.П. (прізвище та ініціали)

Керівник (підпис) Михалик Д.М. (прізвище та ініціали)

Нормоконтроль (підпис) Бойко І.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)

Рецензент (підпис) Луцик Н.С. (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем та програмної інженерії

(повна назва факультету)

Кафедра Кафедра програмної інженерії

(повна назва кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) (прізвище та ініціали)

« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр

(назва освітнього ступеня)

за спеціальністю 121 – Інженерія програмного забезпечення

(шифр і назва спеціальності)

студенту Дрозду Олегу Петровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Веб-клієнт електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua

Керівник роботи Михалик Дмитро Михайлович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	**Примітка**

Студент

(підпис) (прізвище та ініціали)

Керівник роботи

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота «Веб клієнт електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua» Дрозд Олег Петрович, Тернопільський національний технічний університет імені І. Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СПм–61, Тернопіль, 2020.

Пояснювальна записка містить: 82с. 9 рис., 8 табл., 3 дод., 18 бібліогр..

Метою роботи є розробка веб-клієнту для спрощення доступу до електронної почти з підтримкою мобільних пристроїв. В ході роботи досліджено та проаналізовано предметну область, визначено ключових акторів системи, спроектовано ефективну базу даних, застосовано сучасний підхід до розробки, створено зручний дизайн, виконано тестування.

Розроблена система написана на мові програмування JavaScript. Для реалізації інтерфейсу була застосована бібліотека React.

Ключові слова: бібліотека, OAuth2, JavaScript, React, життєвий цикл програмного забезпечення.

SUMMARY

Master thesis "Web-client for tntu.edu.ua email with mobile devices support" Drozd Oleh, Pulyu Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, SPM–61 Group, Ternopil, 2020.

Pages. – 82, pictures. – 9, tables. – 8, add. – 3, bibl.ref. – 18.

The purpose of the work is to develop a web application to simplify access to university emails. In the course of the work the subject area was investigated and analyzed, main system actors were identified, an effective component-based system were designed, a modern approach to development was applied, a convenient design was created.

The system is written in JavaScript programming language. OAuth2 is using for implement authorization in google services. React.js was used to create a responsible and dymanic web pages.

Keywords: Library, OAuth2, JavaScript, React, Softwhere development life cycle.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер, робоча машина для розробки та виконання програм.

ТЗ – технічне завдання

БД – база даних, місце збереження даних

Фреймворк – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та базову функціональність системи або підсистеми.

ОП – охорона праці.

SPA – одно-сторінковий сайт.

ЗМІСТ

ВСТУП

Спочатку те, що люди ділилися через Інтернет, складалося здебільшого із статичної інформації знайдено у файлах. Вони можуть редагувати ці файли та оновлювати їхній вміст, але такі були небагато справді динамічних інформаційних послуг в Інтернеті. Звичайно, їх було небагато винятки: пошук додатків для пошуку файлів, знайдених в архівах FTP та Gopher сервери; та послуги, що надають динамічну інформацію безпосередньо, наприклад, погоду, або наявність банок від машини для видачі соди. (Одне з перших веб програми, які Тим Бернерс-Лі продемонстрував у CERN, були шлюзом для пошуку збільшення номерів з бази даних телефонної книги за допомогою веб-браузера.) Однак здебільшого інформаційні ресурси, якими обмінюються в Інтернеті, були статичні документи. Динамічні інформаційні послуги - від пошукових систем до CGI сценарії до пакетів, які підключали Інтернет до реляційних баз даних - все це змінили. З появою динамічного Інтернету планку підняли ще вище. Не довше чи достатньо було сказати, що ви розробляєте «веб-сайт» (на відміну від строкатого колекція "веб-сторінок"). Виникла необхідність розробити веб-додаток.

Метою магістерської роботи є розробка веб додатку який полегшить доступ до почти tntu.edu.ua.

Для виконання поставлених задач мігістерської роботи та досягнення мети потрібно:

- проаналізувати предметну область
- спроектувати модель програмного забезпечення
- реалізувати програмну систему
- провести тестування

Оскільки переважна більшість людей використовує мобільні пристрої, тому додаток повиннен бути спроектованим та реалізованим з підтримкою мобільних пристроїв.

1 ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

1.1 Аналіз вимог до програмної системи

Статистика розробки програмних проектів передбачає, що більшість збоїв та невдач проекту зумовлені неповними або непослідовними, незрозумілими вимогами, тому аналіз вимог стає ще більш важливішим на цьому етапі розробки програмного забезпечення бере відповідальність за успішне отримання чітких і лаконічних вимог. повні вимоги та вдосконалення системи.

Процес аналізу вимог фокусується на формуванні або отриманні інформації від зацікавлених сторін та інших джерел. Це вимагає вивчення галузі, обговорення основних очікувань програмної системи.

Аналіз вимог - це процес визначення очікувань користувачів щодо програми, яку потрібно створити або змінити. Аналіз вимог включає всі завдання, що виконуються для виявлення потреб різних зацікавлених сторін. Аналіз системних вимог пришвидшує створення програмного продукту та підвищує його надійність та якість.

Заключим етапом визначення вимог є перегляд виконаної роботи по ідентифікації, виявленню та специфікації вимог. На цій стадії зазвичай остаточно узгоджуються увимоги.

1.1.1 Аналіз предметної області

Вибраною темою магістерської роботи є «Розробка веб-клієнта електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua».

Якщо говорити про загальні принципи, які потрібно зрозуміти щоб правильно розробляти ефективні веб-додатки, то ми маємо на увазі основний набір протоколів та мов, пов'язаних із веб-додатками. Це, звичайно, включає HTTP (HyperText Transfer Protocol) і HTML (Мова розмітки HyperText), які є фундаментальними для створення та передачі веб-сторінок. Він також включає старіші Інтернет-протоколи, такі як Telnet та FTP, протоколи, що використовуються для передачі повідомлень, такі як SMTP та IMAP, а також вдосконалені протоколи та мови, такі як XML. Крім того, він включає знання баз даних та мультимедійна презентація, оскільки багато сучасних веб-програм використовують це цих технологій широко. Тому для розробки веб-додатків, ми маєм розуміти не тільки HTTP та HTML, але й інші базові Інтернет-протоколи. Також бути знайомими з JavaScript, XML, JSON, реляційними бази даних, графічним дизайном та мультимедіа. Добре знати технології сервера додатків і мати значний досвід роботи в інформаційній архітектуру.

1.1.1.1 Електронна почта

Електронна пошта була, мабуть, першим «вбивчим додатком». Оскільки мережа сягала корінням військових інтересів, природно, тон електронна пошта почала бути офіційною, жорсткою та діловою. Але раз кількість людей, які користуються електронною поштою, розширилося, і як тільки ці люди зрозуміли, що це може бути раніше використовувалося, речі досить легкі. Електронні списки розсилки надавали громади, де люди з подібними інтересами могли обмінюватися повідомленнями. Ці списки були закритими системами, в тому сенсі, що тільки абоненти могли розміщувати повідомлення в списку або переглядати повідомлення, розміщені іншими передплатниками. Очевидно, що списки зростали, і менеджери списків повинні були їх підтримувати. Через деякий час, були розроблені автоматизовані механізми, що дозволяють людям підписуватися (і, як що важливо, скасувати передплату) без втручання людини. Ці списки розсилки еволюціонували на форуми

повідомлень, де люди могли публічно публікувати повідомлення в електронній мережі дошка оголошень, щоб усі могли прочитати. Ці послуги, безперечно, існували ще до того, як існував Інтернет. Але в ті часи, користувачі читають і відправляють свої електронні листи, безпосередньо входячи в систему (зазвичай через телефонний підключення або пряме місцеве підключення) та запуску програм у цій системі (як правило, з інтерфейсом командного рядка) для доступу до послуг електронної пошти. Методи для використання цих послуг сильно різнилось від системи до системи та підключення до електронної пошти між різними системами важко було знайти. З появою TCP / IP, механізми надання цих послуг стали більш послідовними, а електронна пошта стала рівномірний і всюдисущий. Передача електронної пошти здійснюється за протоколом SMTP. читання електронної пошти зазвичай здійснюється через POP або IMAP.

1.1.1.2 Протокол SMTP

SMTP розшифровується як Простий протокол передачі пошти. Як протокол прикладного рівня, SMTP зазвичай працює поверх TCP, хоча теоретично він може використовувати будь-який базовий транспортний протокол. Додаток під назвою "sendmail" є реалізацією Протокол SMTP для систем UNIX. Остання специфікація протоколу SMTP визначено в Інтернеті RFC 821, а структура повідомлень SMTP визначена в Інтернет RFC 822. SMTP, як і інші служби TCP / IP, працює як сервер, сервіс або демон. В У середовищі TCP / IP SMTP-сервери зазвичай працюють через порт 25. Вони чекають запитів надсилати повідомлення електронної пошти, які можуть надходити від користувачів локальної системи або від через мережу. Вони також відповідають за оцінку адрес одержувачів які знаходяться в повідомленнях електронної пошти та визначають, чи дійсні вони та / або їх кінцевим пунктом призначення є інший одержувач (наприклад, адреса переадресації або набір окремі одержувачі, підписані на список розсилки). Якщо повідомлення, вбудоване у запит, призначене для користувача з обліковим записом у локальній системі, тоді SMTP-сервер доставить повідомлення цьому

користувачеві через додавши його до своєї поштової скриньки. Залежно від реалізації, поштова скринька може будь-що - від простого текстового файлу до складної бази даних електронних повідомлень. Якщо повідомлення призначене для користувача в іншій системі, тоді сервер повинен зрозуміти дізнатися, як передати повідомлення у відповідну систему. Це може передбачати пряме підключення до віддаленої системи або може передбачати підключення до шлюзової системи. Шлюз відповідає за передачу повідомлення на інші шлюзи та / або відправляючи його безпосередньо до кінцевого пункту призначення. До появи SMTP основні механізми надсилання пошти різнились залежно від системи. Одного разу SMTP став повсюдним як механізм при передачі електронної пошти ці механізми стали більш уніфікованими. Програми, відповідальні за передачу повідомлень електронної пошти, такі як SMTP сервери, відомі як MTA (Mail Transfer Agents). Так само, додатки відповідає за отримання повідомлень з поштової скриньки, включаючи сервери POP та Сервери IMAP відомі як MRA (агенти отримання пошти). Клієнтські програми електронної пошти, як правило, були розроблені, щоб дозволити користувачам обидва читати пошту та надсилати пошту. Такі програми відомі як MUA (Mail User Agents). MUA розмовляють з MRA для читання пошти та з MTA для надсилання пошти. У типовому електронному листі клієнт, це процес, за допомогою якого надсилається повідомлення. Після того, як користувач створив повідомлення, клієнтська програма направляє його на SMTP-сервер. По-перше, він повинен підключитися на сервер. Це робиться шляхом відкриття сокета TCP до порту 25 (порт SMTP) сервера. (Це справедливо, навіть якщо сервер працює на машині користувача.)

Клієнтська програма ідентифікує себе (а систему ввімкнено який він працює) на сервер за допомогою команди ‘HELO’. Сервер вирішує (на основі цієї ідентифікаційної інформації) прийняти чи відхилити запит.

Якщо сервер приймає запит, він чекає, поки клієнт надішле додаткову інформацію.

Клієнт по черзі передає команди на сервер, надсилаючи інформацію про ініціатора повідомлення (за допомогою команди ‘MAIL’) та кожну з одержувачів (за допомогою серії команд „RCPT“). Як тільки все це буде зроблено, клієнт повідомляє серверу, що збирається надіслати фактичні дані: саме повідомлення. Це робить до надсилання командного рядка, що складається лише зі слова „ДАНІ”. Кожен наступний рядок, доки сервер не зустріне рядок, що містить лише крапку, вважається частиною тіла повідомлення. Після того, як він надіслав тіло повідомлення, клієнт подає сигнал сервер, що це зроблено, і сервер передає повідомлення за призначенням (або безпосередньо, або через шлюзи). Отримавши підтвердження того, що сервер передав повідомлення, файл клієнт замикає сокетне з'єднання за допомогою команди «QUIT». Спочатку SMTP-сервери виконувались дуже відкрито: кожен, хто знає. Адреса SMTP-сервера може підключатися до нього та надсилати повідомлення. Намагаючись стримувати спам (надсилання невибіркових масових електронних листів у напіванонімних моди), багато реалізацій SMTP-серверів дозволяють системному адміністратору налаштувати сервер так, щоб він приймав з'єднання лише з дискретного набору систем, можливо, лише ті, хто знаходиться в їх локальному домені.

При створенні веб-додатків, що включають функції електронної пошти (зокрема пересилання електронної пошти), переконайтеся, що ваша конфігурація містить специфікацію а працює SMTP-серверна система, яка прийме ваші запити на передачу повідомлень.

Щоб максимізувати гнучкість додатків, адреса SMTP-сервера має бути а параметр, який може бути змінений під час виконання адміністратором програми. Спочатку SMTP-сервери виконувались дуже відкрито: кожен, хто знає Адреса SMTP-сервера може підключатися до нього та надсилати повідомлення. Намагаючись стримувати спам (надсилання невибіркових масових електронних листів у напіванонімних моди), багато реалізацій SMTP-сервера дозволяють системному адміністратору налаштувати сервер так, щоб він приймав з'єднання лише з дискретного набору систем, можливо, лише ті,

хто знаходиться в їх локальному домені. При створенні веб-додатків, що включають функції електронної пошти (зокрема пересилання електронної пошти), переконайтеся, що ваша конфігурація містить специфікацію а працює SMTP-серверна система, яка прийме ваші запити на передачу повідомлень. Щоб максимізувати гнучкість додатків, адреса SMTP-сервера має бути а параметр, який може бути змінений під час виконання адміністратором програми.

1.1.1.3 Протокол FPT

Послуги електронної пошти та обміну повідомленнями в режимі реального часу представляють швидкоплинні, тимчасові комунікації по інтернету. Як тільки миттєве повідомлення чи повідомлення електронної пошти прочитано, воно, як правило, відкидається. Навіть повідомлення на форумі, навіть якщо вони заархівовані, не мають певної міри постійності, і здебільшого тих, хто розміщує такі повідомлення схильні не ставитися до них як до чогось більшого, ніж до перехідних діалогів (або, в деяких випадках справи, монологи). Однак надання віддаленого доступу до більш стійких документів та файлів є принципова необхідність забезпечити спільний доступ до ресурсів. За роки до існування Інтернету файли обмінювались за допомогою BBS (електронні системи дошки оголошень). Люди набирали номер BBS через модем, і після підключення вони матимуть доступ до каталогів файлів для завантаження (а іноді і "скидати" каталоги, до яких можна завантажувати власні файли). Для включення цієї функції по телефону використовувались різні протоколи передачі файлів комутовані лінії (наприклад, Kermit, Xmodem, Zmodem). Щоб полегшити цю функціональність через Інтернет, протокол передачі файлів (FTP) було створено.

FTP-сервер працює аналогічно серверу електронної пошти. Команди існують для автентифікуйте підключеного користувача, надайте йому інформацію про наявні файли та дозволяють користувачеві отримувати вибрані файли. Однак сервери електронної пошти дозволяють вам отримати доступ лише до попередньо встановленої колекції папок (наприклад, вхідних),

виключно для цілей завантаження файлів повідомлень. Сервери FTP також дозволяють користувачам переходити до різних каталоги в локальній файловій системі сервера та (якщо авторизовано) для завантаження файлів до цих каталогів. Специфікація FTP за ці роки пройшла ряд ітерацій, але останню версію можна знайти в Інтернеті RFC 959. У ній описується процес за допомогою якого FTP-сервери роблять файли доступними для клієнтів FTP. Спочатку користувач підключається до FTP-сервера за допомогою клієнтської програми FTP. Для взаємодії FTP зазвичай потрібні два з'єднання між клієнтом і сервером. Один, управління з'єднанням, передає команди та відповіді про стан між клієнтом і сервера. Інший, з'єднання даних, - це зв'язок, по якому фактичні дані відбуваються передачі. Автентифікація користувача відбувається, звичайно, через контрольне з'єднання. Після підключення та автентифікації користувач надсилає команди для встановлення режимів передачі, змінювати каталоги, перераховувати вміст каталогів та передавати файли. Так чи ні користувач може вводити конкретні каталоги, переглядати вміст каталогів, завантажувати файли та / або завантаження файлів залежить від привілеїв безпеки, пов'язаних з його / її обліковим записом користувача на сервері. (Зверніть увагу, що кореневий каталог FTP-сервера не повинен бути таким, як кореневий каталог локальної файлової системи серверного комп'ютера. Системні адміністратори може налаштувати FTP-сервери таким чином, щоб бути доступним лише дискретне піддерево каталогів через FTP-сервер.) FTP-сервери можуть дозволити відкритий доступ до файлів, не вимагаючи явної автентифікації користувача, використовуючи службу, яка називається анонімним FTP. Коли налаштовано FTP-сервер для підтримки анонімного FTP визначається ідентифікатор користувача під назвою «анонімний», який приймає будь-який пароль. „Нетикет" (Інтернет-етикет) передбачає надання користувачами їх електронну адресу як пароль. Системний адміністратор може додатково обмежити піддерево файлової системи, доступне «анонімним» користувачам, як правило, надає доступ лише для читання (хоча можливо налаштувати папку "drop", в яку анонімні

користувачі можуть розміщувати файли). Більшість архівів FTP знайдено в Інтернеті використовувати анонімний FTP для забезпечення відкритого доступу до файлів. Інші протоколи файлових серверів з'явилися протягом багатьох років, але жоден із них не з'явився досягла популярності FTP. З появою наступного покоління розподілених систем обміну файлами, таких як система, яку використовує Napster, ми можемо очікувати змін у на наступних кількох роках.

1.1.2 Постановка задачі

Завданням магістерської роботи є розробка веб клієнту електронної почти який буде пітримувати почту tntu.enu.ua. Програмне забезпечення повинне мати простий та зручний інтерфейс. Програмне забезпечення буде реалізоване з підтримкою мобільних пристроїв відповідати усім рекомендаціям та стандартам W3C.

Після проведення аналізу предметної області визначено наступні етапи розробки:

1) Визначення акторів системи
2) Визначення варіантів використання
3) Вибір моделі розробки
4) Вибір архітектурного патерну
5) Побудова інтерфейсу
6) Звернення до почтового серверу
7) Вибір середовища розробки та мову програмування
8) Огляд основних технологій та фреймворків вибраних для реалізації
9) Реалізація системи

Під час визначення варіантів використання та акторів системи, потрібно описати функціонал системи, розмежувати відповідальності системи. Це

допоможе визначитись з вибором основних технологій, архітектурних патернів, створити інтерфейс та отримувати данні з серверу.

1.1.3 Пошук акторів та варіантів використання

Варіанти використання охоплюють вимоги користувачів до системи, описуючи, як система буде використовуватися та чим закінчується виконання певної послідовності дій, щоб можна було зрозуміти кінцевий результат.

Ознайомившись з поставленою задачею та вимогами було визначено, що в даній системі буде лише один актор – Користувач. Оскільки технічним завданням до системи не передбачено користувачів системи з різним доступом до системи та різним функціоналом.

Користувач – повинен мати змогу проводити розпізнавання предметів, розпізнавання тексту, перегляд, додавання та видалення предметів та тексту.

Після визначення основного актора системи можна приступити до визначення основних варіантів використання системи. Результат виявлення варіантів використання представлено в таблиці 1.1.

Таблиця 1.1 Виявлення варіантів використання

Користувач	Перегляд листів	Можливість переглядати список листів.
Користувач	Перегляд конкретного листа	Можливість переглядати лист від конкретного колистувача
Користувач	Фільтрація листів по даті	Користувач має змогу фільтрувати листи по даті, встановлювати верхній та нижній ліміт
Користувач	Фільтрація листів по адресанту	Користувач отримує змогу фільтрувати листи.
Користувач	Показ непрочитаних повідомлень	Користувач буде бачити свої нерпочитані повідомлення у першу чергу.
Користувач	Доступ до історії повідомлень	Користувач має мати доступ до історії повідомлень
Користувач	Може сортувати повідомлення	Користувач може сортувати повідомення щоб спочатку показувати вибрані.
Користувач	Можливість додати до обраних повідомлень	Система дозволяє додати повідомлення до обраних та переглянути його пізніше у відповідній категорії.
Користувач	Можливість зайти у систему.	Можливість зайти у систему за допомогою електронної почти та пароля.

Користувач	Надсилати повідомлення користувачу	Система надає користувачу можливість надіслати повідомлення іншому користувачу.
Користувач	Надсилання повідомлення групі користувачів	Користувач може надіслати повідомлення групі користувачів.
Користувач	Відповідання на повідомлення	Користувач може відповідати на повідомлення групі осіб
Користувач	Очищення історії повідомлень	Користувач очищає історію повідомлень за вибраний період, що вже відбулись.

Після успішного виявлення варіантів використання системи було побудовано діаграму варіантів використання системи (рисунок 1.1). На даній діаграмі відображено основні варіанти використанні описані вище.

Діаграма варіантів використання - це динамічна діаграма або схема поведінки в реалізована за допомогою уніфікованої мови розмітки UML. Використання діаграми варіантів використання дозволяє змоделювати функціональність системи за допомогою акторів та наданому їм фунціоналу. Варіанти використання в цих діаграмах - це сукупність дій, які потрібно виконувати системі. У цьому контексті "система" - це те, що розробляється чи експлуатується, наприклад, мобільний додаток. "Актори" - це люди, що мають визначену роль в системі.

Діаграми використання є важливими для візуалізації функціональних вимог системи, що перетворюється у вибір моделі, архітектури та засобів розробки програмного забезпечення. Вони також допомагають визначити будь-які внутрішні чи зовнішні фактори, які можуть впливати на систему, і їх слід враховувати. Вони забезпечують хороший аналіз функціональності

системи. Діаграми використання вказують на те, як система взаємодіє з суб'єктами, не уточнюючи як буде реалізована функціональність програмного забезпечення.

1.1.4 Опис ключових варіантів використання

Провівши аналіз варіантів використання системи, можемо визначити самі основні, до них відносяться: редагування профілю, додавання оголошень, перевірка оголошень та оцінка угоди.

Таблиця 1.2 Опис варіанту використання «Вхід у систему»

Дійові особи	Користувач, Система
Цілі	Увійти в систему.
Передумова	Користувач повинен мати адрес електронної почти.
Успішний сценарій: 1. Користувач запускає додаток. 2. Користувач бачить форму авторизації. 3. Користувач вводить усі дані в запропоновані поля та натискає кнопку «Увійти». 4. Додаток перевіряє коректність введених даних. 5. Додаток робить запит до сервера. 6. Відбуваєтсья переадресація на сторінку з листами.	
Результат	Користувач успішно авторизувався.
Альтернативні сценарії	
***а**	Помилка з’єднання з сервером Додаток показує спливаюче повідомлення про немосправність сервера. Результат: вхід у систему не вдається.

4a	Користувач не заповнив поля. Додаток показує спливаюче повідомлення про необхідність заповення полів. Результат: помилка валідації.
4b	Користувач ввів некоректні дані у поля. Додаток показує спливаюче повідомлення про необхідність помилковий формат полів. Результат: помилка сервера.

Таблиця 1.3 Опис варіанту використання «Перегляду повідомлень»

Дійові особи	Користувач, Система
Цілі	Перегляд вхідних повідомлень
Передумова	У користувача є принаймні одне повідомлення
Успішний сценарій: 1. Користувач вводить корректний email та пароль. 2. Користувач авторизуєтсья у системі. 3. Користувач бачить список листів.	
Результат	Користувач успішно отримав доступ до почти.
Альтернативні сценарії	
***a**	Відсутнє підключення до інтернету Додаток показує спливаюче повідомлення про неможливість завантажити повідомлення. Результат: помилка сервера

Таблиця 1.4 Опис варіанту використання «Перегляд історії повідомлень»

Дійові особи	Користувач, Система
Цілі	Перегляд історії повідомлень
Передумова	У користувача є більше 20 повідомлень
Успішний сценарій: 1. Користувач авторизується у системі. 2. Користувач бачить список листів. 3. Користувач може переключатись між сторінками. 4. Додаток робить запит до сервера щоб отримати повідомлення. 5. Додаток відображає інформацію на наступній сторінці.	
Результат	Користувач успішно переглядає історію повідомлень.
Альтернативні сценарії	
***а**	Помилка з’єднання з сервером Додаток показує спливаюче повідомлення про неможливість завантажити повідомлення. Результат: користувач не бачить повідомлень.
4	Користувач має недостатньо повідолень. Додаток показує усі існуючі повідомлення в одній сторінці.

Таблиця 1.5 Опис варіанту використання «Відправка повідомлення»

Дійові особи	Користувач, Система
Цілі	Відправити повідомлення іншому користувачу.
Передумова	Мати електронну адресу отримування.
Успішний сценарій: 1. Користувач натискає кнопку «Написати». 2. Відкривається нове вікно. 3. Користувач вводить необхідну інформацію. 4. Додаток перевіряє коректність введених даних. 5. Додаток відправляє запит з повідомленням на сервер. 6. У додатку з'являється спливаюче повідомлення про успішне надсилання повідомлення.	
Результат	Користувач успішно надіслав повідомлення.
Альтернативні сценарії	
***а**	Відсутнє підключення до інтернету Додаток показує спливаюче повідомлення про неможливість відправити повідомлення. Результат: повідомлення не відправлене.
4	Користувач ввів некоректний адрес електронної пошти отримувача. Додаток показує спливаюче повідомлення про необхідність введеня коректних даних. Результат: повідомлення не відправлене.
5	Виникає помилка на сервері Додаток показує спливаюче повідомлення про неможливість відправити повідомлення. Результат: повідомлення не відправлене.

1.2 Проектування програмної системи

1.2.1 Вибір моделі розробки

Моделі розробки програмного забезпечують численні та придатні для розробки різних типів програмного забезпечення та додатків. Вибір правильної моделі важливий для досягнення очікуваних результатів вчасно та за рахунок бюджету. Навпаки, вибір неправильної моделі або робота без неї може призвести до тривалих термінів виконання, низької якості продукції або прямого провалу проекту.

Тому знання відмінностей між популярними методологіями розробки ПЗ, а також їх плюсів і мінусів є важливим при виборі найбільш оптимальної. Сьогодні ми розглянемо загальні поняття та проведемо таке порівняння, щоб ви могли зважено прийняти рішення, яка модель найкраще відповідає вимогам проекту.

Розглянемо загальне поняття SDLC та декілька найпопулярніших моделей розробки програмного забезпечення, які зараз використовуються: waterfall, iteration model, Kanban.

SDLC - це процес, який застосовується для програмного проекту в рамках організації програмного забезпечення. Це складається з детального плану, що описує, як розробляти, підтримувати, замінювати та змінювати або вдосконалити специфічне програмне забезпечення. Життєвий цикл визначає методологію підвищення якості програмного забезпечення та загального процесу розробки.

Типовий життєвий цикл розробки програмного забезпечення складається з таких етапів:

Етап 1: Планування та аналіз потреб: Найважливішим є аналіз потреб і основний етап у SDLC. Це виконується старшими членами команди з входами від замовника, відділу продажів, опитувань ринку та експертів доменної галузі. Потім ця інформація використовується для планування основного проектного підходу та проведення продукту техніко-економічне

обґрунтування в економічній, експлуатаційній та технічній сферах. Планування вимог до забезпечення якості та виявлення ризиків, пов'язаних із цим проект також виконується на стадії планування. Результатом техніко-економічного обґрунтування є визначити різні технічні підходи, яких можна дотримуватися для реалізації проекту успішно з мінімальними ризиками.

Етап 2: Визначення вимог: Після аналізу вимог наступним кроком є чітко визначити та задокументувати вимоги до продукції та отримати їх затвердження у замовник або аналітики ринку. Це робиться за допомогою «SRS» - вимоги до програмного забезпечення Специфікаційний документ, який складається з усіх вимог до продукту, що підлягають розробці, та розроблений протягом життєвого циклу проекту.

Етап 3: Проектування архітектури продукту: SRS є посиланням для архітекторів продуктів вийти з найкращою архітектурою для продукту, який слід розробити. На основі вимоги, зазначені в ЄСВ, як правило, більше одного підходу до проектування продукту архітектура запропонована та задокументована в DDS - Специфікація проектного документа. Дизайнерський підхід чітко визначає всі архітектурні модулі продукту разом із його представлення зв'язку та подання даних із зовнішніми та сторонніми модулями (якщо будь-який). Внутрішнє проектування всіх модулів запропонованої архітектури повинно бути чітким визначено з найдрібнішими деталями в DDS.

Етап 4: Створення або розробка продукту: На цьому етапі SDLC фактичний розвиток запускається і продукт будується. Програмовий код генерується відповідно до DDS під час цього етап. Якщо проектування виконується детально та організовано, може бути створено код здійснено без особливих клопотів. Розробники повинні дотримуватися вказівок щодо кодування, визначених їх організацією та для генерації коду використовуються інструменти програмування, такі як компілятори, інтерпретатори, налагоджувачі тощо. Для цього використовуються різні мови програмування високого рівня, такі як C, C ++, Pascal, JavaScript та PHP

кодування. Мова програмування вибирається відповідно до типу програмного забезпечення розвинена.

Етап 5: Випробування продукту: Цей етап, як правило, є підмножиною всіх етапів, як на сучасному Моделі SDLC, тестування в основному бере участь на всіх етапах SDLC. Однак це стадія стосується лише стадії випробування товару, де повідомляється про дефекти продукції, відстежуються, фіксуються та повторно випробовуються, поки продукт не досягне стандартів якості, визначених у ЄСВ.

Етап 6: Розгортання на ринку та обслуговування: Після випробування та готовності продукту для розгортання він випускається офіційно на відповідному ринку. Колись продукт розгортання відбувається поетапно відповідно до бізнес-стратегії організації. Товар може вперше випущений обмеженим сегментом і протестований в реальному діловому середовищі (UAT - Користувач приймальне випробування). Потім, на основі відгуків, продукт може бути випущений таким, яким він є, або із запропонованими вдосконалення в цільовому сегменті ринку. Після випуску товару на ринок, його обслуговування здійснюється для існуючої клієнтської бази.

Модель Waterfall - це найдавніший підхід SDLC, який використовувався для програмного забезпечення. Модель водоспаду ілюструє процес розробки програмного забезпечення лінійно послідовний потік; отже, його також називають лінійно-послідовною моделлю життєвого циклу. Це означає, що будь-яка фаза в процесі розробки починається лише в тому випадку, якщо попередня фаза є повна. У моделі водоспаду фази не перекриваються.

Підхід "Водоспад" був першою моделлю SDLC, яка широко використовувалася в розробці програмного забезпечення забезпечити успіх проекту. У підході "Водоспад" весь процес програмного забезпечення розвиток поділяється на окремі фази. У моделі водоспаду, як правило, результат одна фаза діє як вхід для наступної фази послідовно.

Послідовні фази у моделі водоспаду:

Збір та аналіз вимог: Усі можливі вимоги системи, якою повинна бути розроблені фіксуються на цьому етапі та документуються у специфікації вимог.

Проектування системи: у цьому вивчаються технічні вимоги з першої фази готується проект фази та системи. System Design допомагає вказувати обладнання та системних вимог, а також допомагає у визначенні загальної архітектури системи.

Впровадження: За допомогою даних проектування системи, система вперше розроблена в невеликі програми, звані одиницями, які інтегруються на наступному етапі. Кожна одиниця є розроблена та перевірений на функціональність, яка називається модульним тестуванням.

Інтеграція та тестування: усі підрозділи, розроблені на етапі впровадження, є інтегровані в систему після тестування кожного блоку. Після інтеграції всієї системи перевіряється на наявність несправностей та збоїв.

Розгортання системи: Після завершення функціонального та нефункціонального тестування продукт розгортається в середовищі споживача або випускається на ринок.

Технічне обслуговування: У клієнтському середовищі виникають деякі проблеми. Виправити ці випуски випущені. Також для вдосконалення продукту деякі кращі версії звільняються. Технічне обслуговування здійснюється, щоб внести ці зміни у клієнта навколишнє середовище.

Плюси водоспадної моделі:

- Просто і легко зрозуміти і використання.
- Легкий в управлінні завдяки жорсткості моделі - кожна фаза має конкретні результати та огляд процес.
- Фази обробляються і заповнюється по одному.
- Добре працює для невеликих проектів де вимоги дуже добрі зрозумів.

- Чітко визначені етапи.
- Простота в упорядкуванні завдань.
- Процес і результати добре задокументовано.

Мінуси водоспадної моделі:

- Робочого програмного забезпечення не виробляється до пізнього періоду життєвого циклу.
- Високий рівень ризику та невизначеність.
- Не вдала модель для складних і об’єктно-орієнтовані проекти. Бідна модель довго і постійно проектів.
- Не підходить для проектів, де вимоги від помірного до високий ризик зміни. Так що ризик і невизначеність висока при цьому процесі модель.
- Важко виміряти прогрес протягом етапів.
- Не може прийняти переодягання вимоги. Робочого програмного забезпечення не виробляється до кінця життєвого циклу. Налаштування обсягу протягом життєвого циклу може закінчити проект
- Інтеграція здійснюється як "великий вибух" на той самий кінець, який не дозволяє виявлення будь-яких технологічних або вузьке місце у бізнесі або проблеми рано.

Ітераційна модель життєвого циклу не намагається розпочати з повної специфікації вимоги. Натомість розробка починається із зазначення та реалізації лише частини програмне забезпечення, яке потім переглядається з метою виявлення подальших вимог. Потім процес повторюється, створюючи нову версію програмного забезпечення в кінці кожного ітерація моделі. Ітераційний процес починається з простої реалізації підмножини програмного забезпечення вимог та ітеративно вдосконалює нові версії, поки не буде створена повна система реалізовано. На кожній ітерації вносяться модифікації конструкції та нові функціональні можливості додані можливості. Основна ідея цього методу - розробити систему через повторювані цикли (повторювані)

і меншими порціями за раз (поступово). Як і інші моделі SDLC, ітеративний та інкрементальний розвиток має певні особливості додатки в галузі програмного забезпечення.

Ця модель найчастіше використовується в наступному сценарії:

- Вимоги до повної системи чітко визначені та зрозумілі.
- Основні вимоги повинні бути визначені; однак деякі функціональні можливості або запитувані вдосконалення можуть еволюціонувати з часом.
- Існує певний час для обмеження ринку.
- Нова команда використовує та вивчає команду розробників під час роботи над проектом.
- Ресурси з необхідним набором навичок відсутні, і їх планується використовувати далі контрактна основа для конкретних ітерацій.
- Є деякі особливості та цілі високого ризику, які можуть змінитися в майбутньому.

Плюси ітераційної моделі розробки:

- Деякі робочі функції можуть бути розвивався швидко і на початку життєвого циклу.
- Результати отримуються на початку і періодично.
- Паралельний розвиток може бути запланований.
- Прогрес можна виміряти. Менш дорого змінювати обсяг / вимоги.
- Тестування та налагодження під час менша ітерація проста.
- Ризики виявляються та вирішуються під час ітерації; і кожна ітерація є важливим етапом управління. Простіше керувати ризиком - високий ризик частина робиться першою.
- Визначені проблеми, проблеми та ризики від кожного приросту може бути використаний / застосований до наступного збільшення.
- Кращий аналіз ризиків.
- Він підтримує мінливі вимоги.

- Початковий час роботи менше.
- Краще підходить для великих та критично важливих проектів.
- Протягом життєвого циклу програмне забезпечення виробляється рано, що полегшує оцінки та відгуки клієнтів.

Мінуси ітеративного підходу:

- Може знадобитися більше ресурсів.
- Хоча вартість змін менша, але він не дуже підходить для змінних вимог.
- Більше уваги керівництва вимагається.
- Проблеми з архітектурою або дизайном системи може виникнути тому, що не всі вимоги зібрані в початок усього життєвого циклу.
- Визначення приростів може знадобитися визначення повної системи.
- Не підходить для менших проектів.
- Складність управління більше.
- Кінець проекту може бути невідомий що є ризиком.
- Потрібні висококваліфіковані ресурси для аналізу ризиків.
- Прогрес проекту дуже високий залежно від аналізу ризику фаза.

При розробці програмного забезпечення Kanban зазвичай асоціюється з Scrum Board, що відображає робочий процес (а не стандартний To-Do, Busy, Done of Scrum), однак дошка є лише одним із способів візуального контролю, пов'язаного з системою Kanban.

Система Kanban призначена для того, щоб дозволити вам "бачити", що відбувається, стежачи за товаром (або послугою) від початку до кінця по всій системі. Це може бути використання кольору (наприклад, зеленого, бурштинового, червоного) для позначення статусу; це може бути картка іншого розміру для позначення кінця партії (або спринту); це може бути число, яке вказує на обмеження незавершеного виробництва. Ключовим є те, що це форма візуального спілкування, яка дозволяє проактивно вживати заходів до того, як ризик стане проблемою. Основна відмінність Kanban від Scrum Board

полягає в тому, що вона підтримує весь життєвий цикл (Kanban), тоді як Scrum Board розглядає лише одну частину системи як чорний ящик.

Плюси Канбана:

- Більш цілісний погляд на весь процес.
- Підтримка прийняття рішень за допомогою системи раннього попередження.
- Більш гнучкі та плавні, змінюючи візуальні показники, коли проблеми / робочі процеси змінюються.

Мінуси Канбана:

- Застаріла дошка Kanban може призвести до проблем у процесі розробки.
- Іноді команда Kanban робить дошку занадто складною.
- Відсутність хронометражу є ще одним недоліком, оскільки відсутні терміни, пов’язані з кожною фазою.

У результаті оцінки переваг та недоліків вище перечислених методологій розробки програмного забезпечення було обрано методологію канбан, адже вона найкраще ілюструє виконані та завданні, що очікують виконання. Дозволяє у графічному відображені показати статус завдання, має зручний менеджмент усіх завдань.

1.2.2 Вибір архітектурного паттерна

Оскільки ми розробляємо додаток під Web, потрібно врахувати особливості браузера та вибрати найбільш оптимальний архітектурний патерн розробки для вирішення поставленої задачі.

Архітектурний паттерн - це багаторазове рішення, яке можна застосувати до загальнопоширених проблем у розробці програмного забезпечення - у нашому випадку - у написанні програм на основі JavaScript. Інший спосіб розгляду шаблонів - це шаблони для вирішення проблем - тих, які можна використовувати у досить багатьох різних ситуаціях. Отже, чому

важливо розуміти закономірності та бути з ними знайомими ?. Дизайн паттерни мають три основні переваги:

1. Шаблони - це перевірені рішення: вони забезпечують надійні підходи до вирішення питань при розробці програмного забезпечення з використанням перевірених рішень, що відображають досвід і ідеї розробників, які допомогли їх визначити та вдосконалити, щоб привести їх до шаблону.

2. Шаблони можна легко використати повторно: шаблон, як правило, відображає нестандартне рішення які можна адаптувати до власних потреб. Ця особливість робить їх досить надійними.

3. Шаблони можуть бути виразними: коли ви дивитесь на шаблон, зазвичай існує набір структура та «словниковий запас» представленого рішення, які можуть скоріше допомогти висловитись великі рішення досить елегантно. Шаблони не є точним рішенням. Важливо пам’ятати про роль шаблону полягає лише у наданні нам схеми рішення. Шаблони не вирішують усіх дизайнерських проблем вони не замінюють хороших дизайнерів програмного забезпечення, проте підтримують їх. Далі ми розглянемо деякі інші переваги, які можуть запропонувати моделі.

Повторне використання шаблонів допомагає запобігти незначним проблемам, які можуть спричинити серйозні проблеми в процесі розробки додатків. Що це означає, коли код побудований на перевірених зразках, ми можемо дозволити собі витрачати менше часу на турботи структуру нашого коду та більше часу, орієнтуючись на якість нашого загального рішення. Це тому, що шаблони можуть спонукати нас до більш структурованого кодування та організована мода, уникаючи необхідності переробляти її з метою чистоти в майбутнє.

Шаблони можуть надати узагальнені рішення, які задокументовані таким чином, що не вимагають прив'язки їх до певної проблеми. Це узагальнено підхід означає, що незалежно від програми, з якою ви працюєте, можна застосувати шаблони дизайну для поліпшення структури вашого коду.

Деякі шаблони насправді можуть зменшити загальний розмір файлу вашого файлу коду, уникаючи повторень. Заохочуючи розробників уважніше придивитися їх рішення для областей, де можна негайно зменшити кількість повторень, наприклад зменшення кількості функцій, що виконують подібні процеси, на користь однієї узагальненої функції, загальний розмір вашої кодової бази можна зменшити.

Шаблони додають до словника розробників, що робить спілкування швидше.

Часто використовувані візерунки можна покращити з часом, використовуючи їх колективний досвід інших розробників, які використовують ці шаблони, сприяють повернутися до спільноти дизайнерських зразків. У деяких випадках це призводить до створення цілком нових дизайнерських зразків, тоді як в інших це може призвести до надання вдосконалених вказівок щодо того, як найкраще використовувати конкретні зразки. Це може гарантувати, що шаблонні рішення продовжують ставати більш надійними, ніж спеціальні рішення може бути.

Розглянемо декілька популярних шаблонів та виберем найбільш оптиальний для нашого програмного рішення:

MVC - це модель архітектурного дизайну, яка заохочує вдосконалення організації заявок шляхом розділення питань. Це забезпечує ізоляцію бізнес-даних (Моделі) з користувальницьких інтерфейсів (Перегляди), з третім компонентом (Контролери) (традиційно), що управляє логікою, вводом користувача та координує як моделі, так і види. візерунок був спочатку розроблений Тригве Ренскаугом під час його роботи над Smalltalk-80 (1979), де його спочатку називали Model-View-Controller-Editor. MVC далі він був детально описаний у "Шаблонах дизайну: Елементи багаторазового об'єктно-орієнтованого програмного забезпечення" ("GoF") у 1994 р., що зіграло роль у популяризації його використання.

У наш час шаблон MVC був застосований до різноманітного діапазону мов програмування, включаючи найбільш важливе для нас: JavaScript. JavaScript тепер має ряд фреймворків похвалившись підтримкою MVC (або її варіаціями, які ми називаємо сімейством MV *), дозволяючи розробникам легко додавати структуру до своїх додатків без великих зусиль. Ви, напевно, стикалися з принаймні одним із таких фреймворків, але вони включають як Backbone, Ember.js та JavaScriptMVC. Враховуючи важливість уникання "спагетті" - термін, що описує код, який дуже важко прочитати чи підтримати через відсутність структури вкрай важливо, щоб сучасний розробник JavaScript розумів, що забезпечує цей шаблон. Це дозволяє нам ефективно оцінювати те, що це фреймворки дозволяють нам робити інакше.

MVC складається з трьох основних компонентів:

- Моделі керують даними програми. Вони не стосуються ні інтерфейсу користувача, ні шарів презентації, а натомість представляють унікальні форми даних, які додатку може знадобитися. Коли модель змінюється (наприклад, коли вона оновлюється), вона зазвичай змінюється повідомляти своїх спостерігачів (наприклад, погляди, концепція, яку ми розглянемо найближчим часом) про те, що відбулися зміни, щоб вони могли відповідним чином відреагувати. Щоб зрозуміти моделі далі, уявімо, що у нас є додаток фотогалереї JavaScript. У фотогалереї концепція фотографії заслуговує на власну модель представляє унікальний тип даних, характерних для домену. Така модель може містити суміжні такі атрибути, як підпис, джерело зображення та додаткові метадані. Конкретна фотографія буде зберігатися в екземплярі моделі, і модель також може бути багаторазовим використанням.

Якщо ви прочитаєте будь-який із старих текстів на MVC, ви можете натрапити на опис моделей, а також управління "станом" програми. У додатках JavaScript "стан" має різне значення, як правило, посилання на поточний "стан", тобто перегляд або допогляд (з конкретні дані) на екрані користувачів у фіксовану точку. Держава - це тема, яка регулярно обговорюється при перегляді

односторінкових додатків, де поняття держави потребує бути змодельованим. Отже, підсумовуючи, моделі в першу чергу стосуються ділових даних.

- Представлення - це візуальне представлення моделей, які представляють відфільтрований вигляд їх поточного держава. Представлення зазвичай спостерігає модель і отримує повідомлення про зміну моделі, дозволяючи поданню відповідно оновлюватись. У літературі про шаблони дизайну зазвичай посилаються розглядати як "німі", оскільки їх знання моделей та контролерів у програмі обмежені. Користувачі можуть взаємодіяти з поданнями, і це включає можливість читання та редагування (тобто отримати або встановити значення атрибутів у моделях). Оскільки вигляд є презентаційним шаром, ми загалом представляють можливість редагування та оновлення в зручному для користувача режимі.

- Контролери - це посередник між моделями та представленнями, які класично відповідають за два завдання: обидва вони оновлюють вигляд, коли модель змінюється, і оновлюються модель, коли користувач маніпулює видом. З точки зору того, де більшість фреймворків JavaScript MVC погіршує те, що прийнято вважати "MVC", проте це стосується контролерів. Причини цього різні, але в моя чесна думка полягає в тому, що автори фреймворку спочатку дивляться на серверну інтерпретацію MVC, розуміють, що вона не перекладає 1: 1 на стороні клієнта і повторно інтерпретує C у MVC означати те, що вони відчувають, має більше сенсу. Проблема з цим однак полягає в тому, що він суб'єктивний, збільшує складність в обох розумінях класичний шаблон MVC і, звичайно, роль контролерів у сучасних рамках. Як приклад, давайте коротко розглянемо архітектуру популярного архітектурного фреймворку Backbone.js. Магістраль містить моделі та подання (дещо схожі на те, що ми розглядали раніше), однак насправді він не має справжніх контролерів. Його погляди і Маршрутизатори діють трохи схоже на контролер, але насправді вони не є контролерами власний. У цьому відношенні, всупереч тому, що може згадуватися в офіційній документації або у повідомленнях в блозі Backbone не є справді MVC / MVP, ані MVVM. Це в Насправді краще вважати його членом

сімейства MV *, який підходить до архітектури по-своєму. Звичайно, у цьому немає нічого поганого, але це важливо розрізняйте класичний MVC та MV *, якщо ви покладаєтесь на поради класична література про перше, щоб допомогти другому.

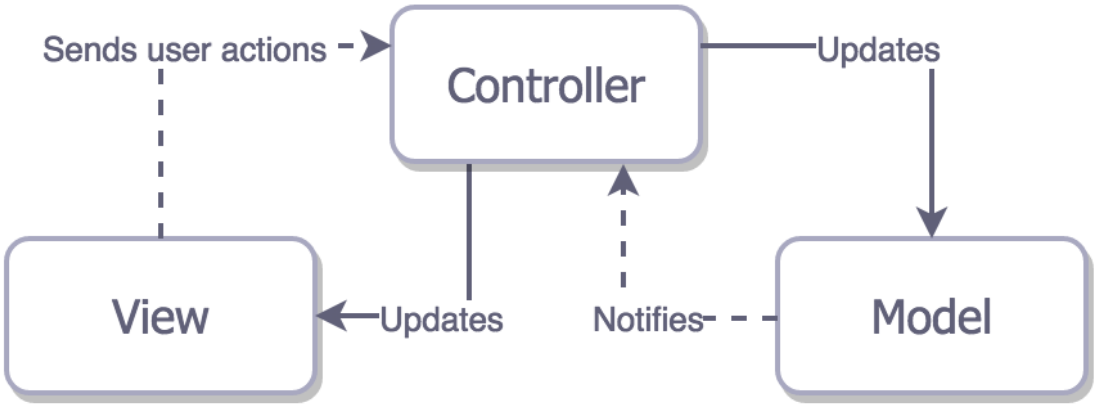

Рисунок 1.4 Схематичне зображення роботи MVC патерну.

Що дає нам MVC? Це розділення проблем у MVC сприяє спрощенню модуляризації функціональних можливостей програми та дозволяє:

• Простіше загальне обслуговування. Коли потрібно зробити оновлення програми дуже ясно, чи ці зміни орієнтовані на дані, тобто зміни моделей та можливо контролери, або просто візуальні, що означає зміни у поданнях.

• Роз'єднання моделей та поглядів означає, що це набагато прямолінійніше написати модульні тести для ділової логіки

• Дублювання моделі низького рівня та коду контролера (тобто, яким ви могли бути замість цього) виключається з усієї програми

• Ця модульність залежить від розміру заявки та розподілу ролей дозволяє розробникам, відповідальним за основну логіку, і розробникам, що працюють над користувальницькими інтерфейсами, працювати одночасно

- **MVVM** (Model View ViewModel) - це архітектурний шаблон, заснований на MVC та MVP, який намагається чіткіше відокремити розробку інтерфейсів користувача (UI) від бізнес-логіки та поведінки в додатку. З цією метою багато реалізацій цього шаблону використовують декларативні прив’язки даних, щоб дозволити поділ роботи над поданнями з інших шарів. Це полегшує роботу з інтерфейсом та розробкою, що відбувається майже одночасно в межах та сама кодова база. Розробники інтерфейсу користувача записують прив'язки до ViewModel у своєму документі розмітки (HTML), де Модель і ViewModel підтримуються розробниками працює над логікою програми.

View та ViewModels спілкуються за допомогою прив’язки даних та подій. Як ми бачили в наш початковий приклад ViewModel, ViewModel не просто виставляє атрибути Model але також доступ до інших методів та функцій, таких як перевірка. Наші подання обробляють власні події в інтерфейсі користувача, відображаючи їх у ViewModel як необхідний. Моделі та атрибути на ViewModel синхронізуються та оновлюються через двостороння прив'язка даних. Тригери (дані-тригери) також дозволяють нам далі реагувати на зміни в стані нашої Моделі атрибути.

Переваги:

• MVVM сприяє спрощенню паралельної розробки інтерфейсу користувача та будівельних блоків.

• Абстрагує подання і тим самим зменшує кількість ділової логіки (або клею), необхідну в коді, що стоїть за ним

• ViewModel може бути простішим для модульного тестування, ніж код, керований подіями

• ViewModel (будучи більше моделлю, ніж View) можна протестувати без побоювань Автоматизація та взаємодія інтерфейсу

Недоліки:

• Для більш простих користувацьких інтерфейсів MVVM може бути надмірним

• Хоча прив’язка даних може бути декларативною та приємною для роботи, вона може бути складнішою для налагодження, ніж імперативний код, де ми просто встановлюємо точки зупинки

• Прив'язка даних у нетривіальних додатках може створити багато проблем.

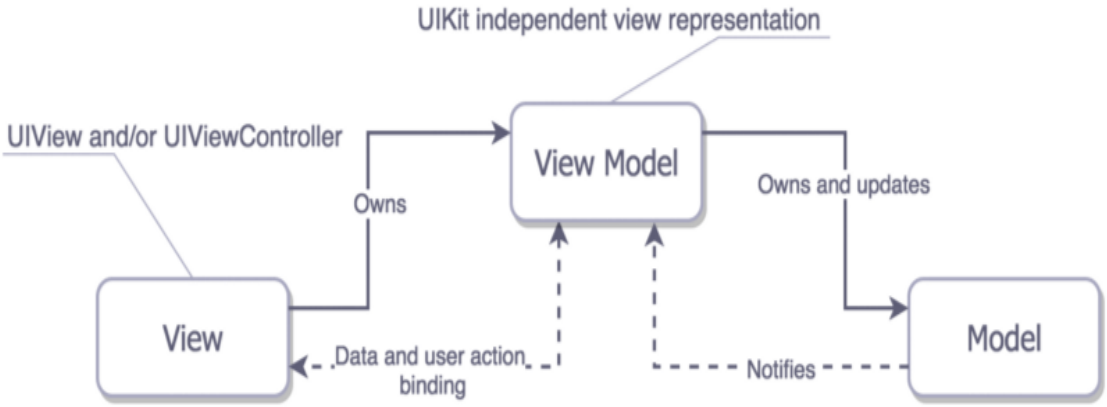

Рисунок 1.5 Схематичне зображення роботи MVVM патерну

Оцінивши усі плюси та недоліки реалізації патернів вирішено використовувати патерн MVC, адже він простий в реалізації і дозволить нам швидко побудувати додаток. У цьому патерні React буде виступати у ролі View.

2 ДОСЛІДЖЕННЯ ПРОГРАМНОЇ СИСТЕМИ

2.1 Вимоги до системи

Для вирішення поставленої задачі потрібно розробити надійний механізм авторизації у почтовій системі для того щоб могти взаємодіяти з програмою. Також це дозволить нам бути впевненим що система не підвергнеться нападу через якусь вразливість у системі.

Тому було вирішено дослідити авторизацію через електронну почту і розробити власне рішення яке дозволить авторизуватись у системі через почту «tntu.edu.ua».

Так як почта «tntu.edu.ua» адмініструється сервісами Google нам потрібно інтегрувати нашу систему з ними і таким чином буде підтримуватись надійна авторизація у програмну систему.

Так як наша ціль це досягти найбільшого захисту від вразливостей системи найкрашим рішенням буде реалізувати авторизацію з використанням протоколу OAuth.

OAuth означає відкриту авторизацію. Це безкоштовний і відкритий протокол, побудований за стандартами IETF та ліцензії від Open Web Foundation, і є правильним рішенням для забезпечення відкритості платформи. Розробники OAuth вирішили вирішити проблему, якої не мають служби та паролі, коли ви починаєте комбінувати програми та затирати їх. Уявіть собі, в сучасному середовищі веб-API та мобільних додатків, якщо кожен веб-сайт, який використовував API з іншого веб-сайту повинен був надавати та зберігати цей пароль веб-сайту. Незабаром ви б забезпечити розповсюдження ваших паролів по всьому Інтернету з будь-якою послугою, якою ви користуєтесь використовується з Facebook, Twitter, Skype, вашого банківського рахунку. Чи довіряєте ви їм усім Ваш пароль? OAuth - це ще один спосіб автентифікації служби - протокол безпеки, який дозволяє користувачам

надавати стороннім доступ до своїх веб-ресурсів, не ділячись своїми паролями. Основою OAuth є маркер авторизації з обмеженими правами, який користувач може скасувати у будь-який час вони можуть стати підозрілими або незадоволеними додатком, для якого використовують отримати доступ до вашого бізнесу. Метафора ключа камердинера OAuth підтримує цю «делеговану автентифікацію» між веб-програмами за допомогою маркера безпеки називається "маркером доступу". Замість того, щоб покладатися на один пароль як головний ключ для кожна програма, яка отримує доступ до API, OAuth використовує цей маркер. Токен OAuth дає доступ одному додатку одному API від імені одного користувача. Еран Хаммер-Лахав, автор специфікації для OAuth, порівнює маркер із ключем камердинера. Це влучна метафора. Для більшості людей їх машина - одне з найцінніших надбань, яке оцінюється десятками тисячі доларів. Вони є зручними місцями, де можна залишити інші наші цінні речі комп'ютери та одяг. Однак іноді від нас вимагають, щоб вони дали їх на стоянку обслуговуючого персоналу чи камердинера, яких ми раніше не зустрічали. Ключ камердинера вирішує проблему - це маркер доступу з обмеженими правами, який може керувати транспортним засобом, але не надає доступ до багажника.

Чому OAuth корисний для провайдерів API?

У попередніх розділах ми говорили про те, чому OAuth корисний для користувачів - як OAuth дозволяє користувачам надати стороннім доступ до своїх веб-сервісів або мобільних додатків, не надаючи їм спільного доступу паролі. Ми вважаємо, що OAuth також корисний для постачальників API, незалежно від того, чи піддають вони їх API веб-додатки або API, призначені для мобільних додатків. OAuth означає, що веб-додатки, що містять API, не мають спільного доступу до паролів. Є дві альтернативи: По-перше, ризик безпеки введення пароля в кожен один веб-додаток, яким ви користуєтесь - неприйнятний ризик! По-друге, якийсь універсальний механізм єдиного входу

(керований третьою стороною, про яку всі погоджуються та довіряють), що, звичайно, не існує! Тому для всіх веб-додатків, які надають API іншим веб-програмам, ми рекомендуємо OAuth. OAuth усуває необхідність зберігати пароль на мобільному пристрої, додаючи шар безпека, коли API використовується мобільними програмами, створеними ненадійними розробниками для загальнодоступного API. Крім того, важко вгадати маркер OAuth (оскільки він набагато довший і випадковіший за пароль.) Він прив'язаний до певної програми та пристрою. Його можна скасувати, не впливаючи інші пристрої та програми. OAuth дозволяє провайдеру API відкликати маркери для окремого користувача для цілого додатка, не вимагаючи від користувача змінити оригінальний пароль. Це критично, якщо мобільний телефон пристрій порушено або якщо виявлено несанкціоновану програму. OAuth також підтверджує процес автентифікації в майбутньому. Оскільки браузери користувачів є перенаправлено в місце, яке знаходиться під контролем команди API, щоб отримати маркер OAuth, Команда API може контролювати, що робити в цей момент робочого процесу. Іншими словами, OAuth є досить гнучким, щоб підтримувати будь-яку специфіку вашої компанії робочі процеси (подумайте про багатофакторну автентифікацію, умови обслуговування, зміни умов обслуговування, і так далі ...), не вимагаючи зміни клієнта або сервера. Тому для API, призначених для мобільних та власних додатків, ми рекомендуємо OAuth.

Роль OAuth для безпеки API?

OAuth став найкращою практикою і є важливим для забезпечення безпечного доступу користувачів та забезпечуючи безперебійну роботу користувачів. Ми настійно рекомендуємо OAuth для провайдерів API коли вони виставляють API для веб- або мобільних додатків, але є кілька сценаріїв який OAuth може бути надмірним чи не найкращим рішенням.

Для API між серверами - API, призначені для використання лише невеликою кількістю серверів - OAuth є надмірним. Наявність окремого набору даних автентифікації для кожної програми - це приємно особливість OAuth, але для використання між серверами необхідність безпечного входу за допомогою браузера, або реалізувати інші кроки в "танці" OAuth, що заважає. Натомість, використовуючи простий стандарт безпеки, такий як автентифікація HTTP та присвоєння достатньо унікального пароля для кожної програми.

Двосторонній SSL - це ще одна користь, хоча громіздкий підхід, який має перевагу сильнішої, більш простежуваної автентифікації. Однак думайте заздалегідь! Чи справді ці API будуть використовуватися серверами лише назавжди? В в майбутньому, можливо, ними будуть користуватися веб-програми чи мобільні клієнти, або кількість серверів зросте далеко за межі того, що ви спочатку уявили. Крім того, OAuth 2.0 має ряд способи безпечного отримання серверами маркерів, які не потребують входу в браузер. OAuth сьогодні це може здатися надмірним, але через кілька місяців це виявляється абсолютно критичним.

Використовуйте SSL для всього чутливого. Якщо тільки ваш API не має відкритих та нечутливих даних даних, підтримувати SSL і розглянути можливість його застосування шляхом перенаправлення будь-якого трафіку API на не-SSL порт до порту SSL. Це робить інші схеми автентифікації більш безпечними та зберігає ваші дані приватного API користувача від сторонніх поглядів - і це не все так складно зробити. Використовуйте ключі API лише для нечутливих даних, доступних лише для читання. Якщо у вас є загальнодоступний API - який виставляє дані, які ви все одно опублікуєте в Інтернеті - розгляньте можливість випуску нечутливого API клавіші. Вони прості у впровадженні і все ще дають вам спосіб ідентифікувати програми та розробники. Озброївшись ключем API, у вас є можливість встановити квоту

для вашого API або, принаймні, моніторинг використання додатком. Без нього єдиним способом відстеження використання є через IP-адреси, які не є надійними. Наприклад, API геокодування Yahoo Maps видає ключі API, щоб він міг відстежувати своїх користувачів і встановлювати квоту, але дані, які він повертає, є не чутливий, тому не критично зберігати ключ у таємниці. ("Двоногий OAuth" - це ще один спосіб для цього захищає ключ API у мережі, не вимагаючи SSL.)

Перевіряйте вхідні та вихідні дані. Це запобіжить зловмисному коду або вмісту введення вашої системи або виконання клієнтами, які читають ваші дані. Ця практика повинна використовувати для всіх API, але може бути особливо важливим для записуваних. Наприклад, для записуваного API ви не хочете дозволити вставку JavaScript, для якого можна використовувати атаки між сценаріями між сайтами, коли користувачі відвідують ваш веб-сайт. Наприклад, якщо параметр Як відомо, числове значення має бути перевірено як числове перед передачею. Знову ж таки, ця практика повинна застосовуватися до всіх API, але може бути більш необхідною для API, що записуються. Чи існують інші сценарії, для яких OAuth є непотрібним або навіть поганою ідеєю? Як і сценарій між серверами, ми вважаємо, що OAuth не має сенсу в наступні сценарії:

- Все, що вимагає комерційного рівня довіри. Наприклад, коли ваша безпека модель вимагає можливостей інфраструктури PKI. Цифровий підпис кожного виклику API повільний і громіздкий, але якщо ваш бізнес цього вимагає, тоді заміни немає.
- Одноразові жетони. OAuth - це багато складності та механізмів для здійснення одного виклику API.

Чи існують інші анти-шаблони OAuth? Погані ідеї включають створення власної "версії" OAuth або створення подібного OAuth але різний.

Дотримуючись стандартів, і зосередьте свої зусилля на розробці створення чудових додатків здається кращою ідеєю, ніж створення власної схеми безпеки.

Отже після вибору та дослідження одного з провайдері авторизації OAuth, розглянемо на практиці як інтегрувати його з веб-клієнтом.

2.2 Розробка концепції системи

Цей потік OAuth 2.0 називається неявним потоком надання. Він призначений для програм, які отримують доступ до API лише тоді, коли користувач присутній у програмі. Ці програми не можуть зберігати конфіденційну інформацію.

У цьому потоці ваш додаток відкриває URL-адресу Google, яка використовує параметри запиту для ідентифікації вашої програми та типу доступу до API, який потрібен додатку. Ви можете відкрити URL-адресу у поточному вікні браузера або у спливаючому вікні. Користувач може пройти автентифікацію в Google і надати потрібні дозволи. Потім Google переспрямовує користувача назад до вашого додатка. Переспрямування включає маркер доступу, який ваш додаток перевіряє, а потім використовує для надсилання запитів API.

Щоб додати авторизацію до проекту потрібно:

1. Мати існуючий аккаунт почти Google.
2. Зайти в Console розробника в сервісах Google.
3. Натиснути кнопку «Створити проект».
4. Дати йому назву і назву організації.
5. Натиснути «Створити».

Далі потрібно для цього проету додати Ідентифікатор Клієнта OAuth, для цього у лівій панелі вибираєм «Учетние данние», «Создать», та з випадаючого списка вибираємо «Ідентифікатор клієнта OAuth», заповнюємо данні та на виході отримуємо ключі нашого проекту які знадобляться для інтеграції з веб клієнтом.

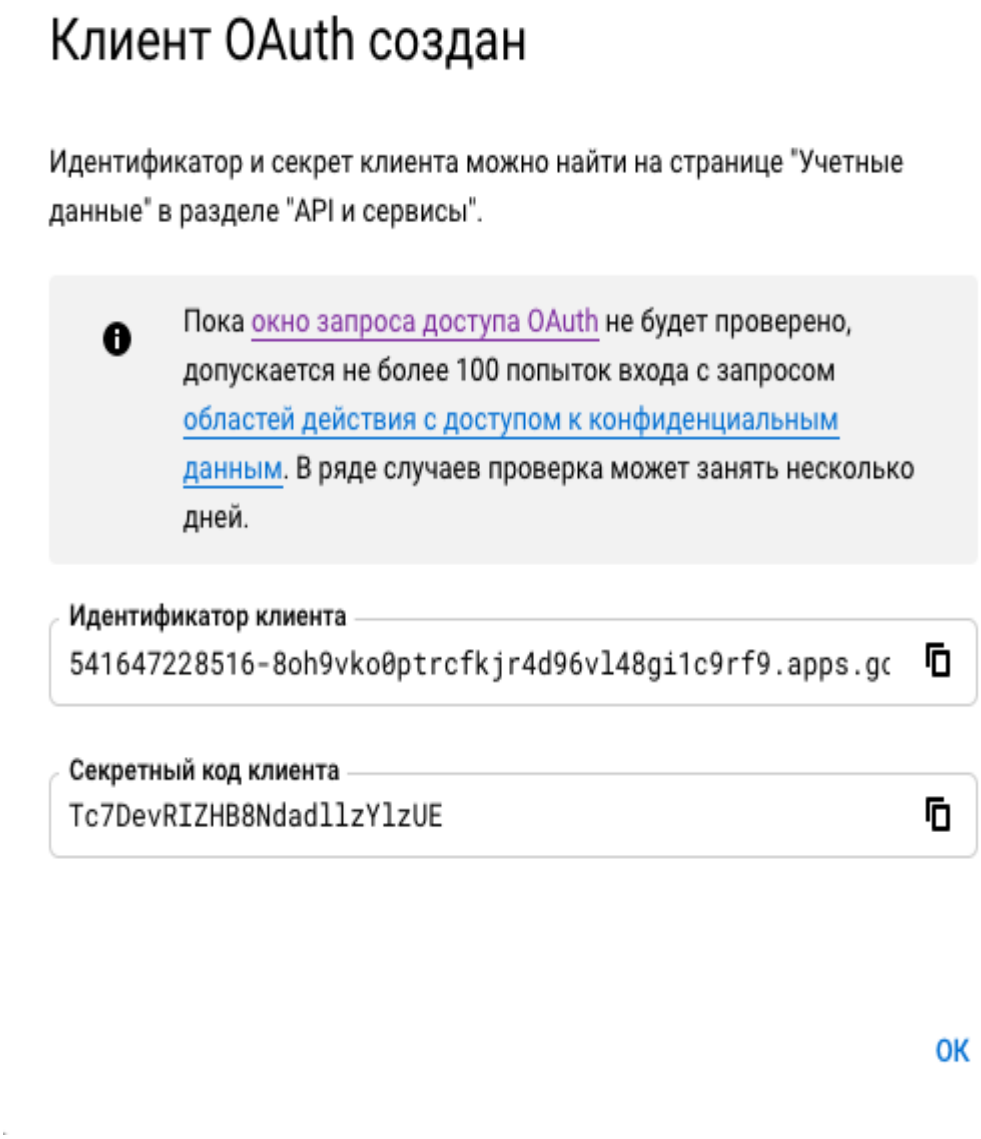

Рисунок 2.1 Створення клієнту OAuth

Зберігаємо Ідентифікатор і секретний код клієнта для подальшого використання.

Та на сторінці має зявитись наш клієнт в списку. Тепер ми можемо користуватись сервісами Google для авторизації, залишилось увікнути

потрібне API щоб потім змогти отримати доступ до листів та до користувачів за допомогою API.

Для цього заходимо в «Панель керування» та переходимо в «Бібліотеку API», в якій потрібно знайти Gmail API.

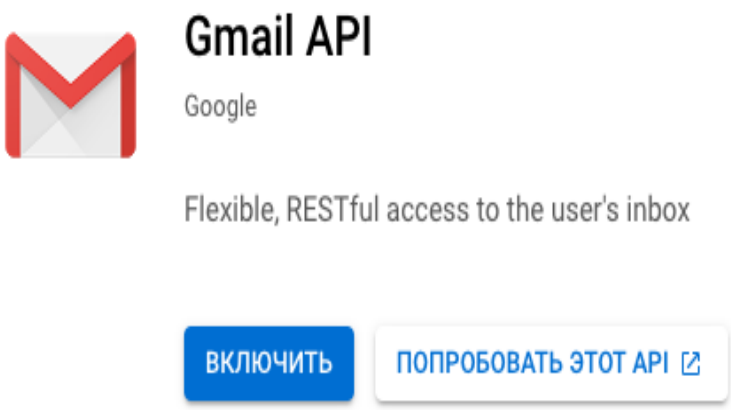

Рисунок 2.2 Увімкнення Gmail API.

Та увікмнути його, далі прочитати угоду та прийняти її. Тепер ми можемо отримувати доступ до почти з нашого проекту. За допомогою ключів які ми створили раніше. Це всі налаштування наразі які нам потрібні для інтеграції OAuth авторизації з Gmail API.

У наступному розділі розглянемо отримання access_token в програмній реалізації, потрібно буде виконати кілька кроків і написати декілька функцій які дозволять нам авторизуватись через OAuth2. В Google є певний flow цієї авторизації якого треба притримуватись аби успішно авторизуватись, далі ми розглянемо імплементацію у веб-клієнті за допомогою мови прогрмування JavaScript.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

Щоб розрозка веб-клієету була корисною та успішною, вибір правильного набору технологій є дуже важливим. Технологічний стек не лише збільшує термін служби проекту, але також збільшує рентабельність, масштабованість та відповідає функціональним вимогам. Відповідний стек технологій може призвести до зниження витрат і зменшення часу на розробку додатків. Тому потрібно вибрати найпотрібніші та найпопулярніші підходи до розробки веб додатків та передові технології, які можна використовувати для їх реалізації.

Розробка програми означає вибір та використання мов програмування, наборів для розробки ПЗ, середовищ розробки та інших інструментів, наданих операційною системою. Як результат, для створення веб додатків для усіх платформ потрібно використовувати окремі технології розробки.

3.1 Вибір технологій розробки

Оскільки система має розроблятись для веб-платформи нам потрібно вибрати мову програмування яка буде підтримуватись браузером. Раніше, коли всі браузери мали різне ядро вони могли по різному інтерпретувати мови програмування, але у наш час це все стандартизовано і якщо побудувати веб-сторінку яка буде працювати на одному бразузері, можна зробити висновок що вона буде працювати так само добре на інших.

Для розробки клієнських додатків існує не великий вибір технологій, тому ми не зможемо розглянути кожну і зупинимось на базовому наборі технології для розробки веб додатків, а саме:

- HTML – мова розмітки гіпертексту. Це основа веб сторінки, за допомогою цієї мови будується каркас сторінки яку ми можем бачити у браузері. Ця мова вважається імперативною бо вона містить багато тегів за допомогою яких можна побудувати розмітку сторінки.

- CSS – це каскадна таблиця стилів для веб сторінки. Вона призначена для того щоб стилізувати веб-сторінку, додати розміщення елементів та різні стилі для них. Також можна задати шрифти та різні анімації які мають відбуватись при певних обставинах.
- JavaScript – щоб зробити нашу сторінку динамічною потрібно додати різні скрипти які будуть спрацьовувати при певних обставинах. Наприклад при натисканні на кнопку «Написати» буде відкриватись вікно з новим повідомленням. Щоб оживити сторінку і зробити її динамічно нам допоможе ця мова.
- React – бібліотека для розробки веб-компонентів з SPA підходом, яку заснувала компанія Facebook.
- Google API, з яким ми вже познайомились у попередньому розділі. Буде використовуватись для отримання доступу до листів та до загального функціоналу.

Також ми розглянемо та використаємо новий принцип побудови веб додатків під назвою Single Page Application (SPA).

У SPA весь додаток працює як одна веб-сторінка. У цьому підході рівень презентації для всієї програми виведено з сервера і є керується з браузера. Завдяки такому дизайну кожен запит на новий вигляд (HTML-сторінка) призводить до зворотного туру на сервер. Коли потрібні свіжі дані на стороні клієнта, запит надсилається до сторона сервера. На стороні сервера запит перехоплюється об'єктом контролера всередині презентаційного шару. Потім контролер взаємодіє з рівнем моделі через сервісний рівень, який визначає компоненти, необхідні для завершення рівня моделі завдання. Після отримання даних або об’єктом доступу до даних (DAO), або службою агента, будь-які необхідні зміни до даних вноситься бізнес-логікою в діловий рівень. Елемент управління передається назад на рівень презентації, де вибирається відповідний вигляд. Логіка презентації диктує, як свіжоотримані дані відображаються в вибраний вигляд. Часто отриманий вигляд починається як вихідний файл із заповнювачами, де потрібно вставити дані (і, можливо, інші

інструкції щодо візуалізації). Цей файл діє як різновид шаблону для того, як подання штампується, коли контролер направляє на нього запит. Після об’єднання даних та подання подання повертається до браузера. Потім браузер отримує нову сторінку HTML, і за допомогою оновлення інтерфейсу користувач бачить нову подання, що містить запитувані дані.

Загальний дизайн SPA майже такий самий, як і традиційний дизайн. Ключ Зміни полягають у наступному: повний браузер не оновлюється, логіка презентації знаходиться в транзакції клієнта та сервера можуть бути лише даними, залежно від ваших уподобань візуалізація даних.

Отже ми розглянули список технологій які будем застосовувати при розробці веб-клієнта, була обрана бібліотека React тому що у ній є Virtual DOM, який значно спростить взаємодію з реальним DOM і привнесе оптимізації необхідні для того щоб веб сторінка працювала і отримувала всі потрібні данні без перезавантаження. Також серед переваг цієї бібліотеки можна відмітити підтримку Facebook і регулярні оновлення, тому можна бути впевненим що у нашому додатку не буде ніяких вразливостей.

3.2 Реалізація авторизації в Google через OAuth2

Щоб реалізувати авторизацію у сервісах Google нам треба буде слідувати інструкції яка представлена у оффіційній документації, також треба буде завантажити бібліотеку за допомогою якої буде відбуватись спілкування з віддаленим API.

Розглянемо базові шляхи імплементації авторизації на стороні клієнта:

Усі додатки дотримуються базової моделі при доступі до API Google за допомогою OAuth 2.0. На високому рівні ви виконуєте п’ять кроків:

1. Отримайте облікові дані OAuth 2.0 із консолі Google API. Відвідайте консоль Google API, щоб отримати облікові дані OAuth 2.0, такі як ідентифікатор клієнта та секрет клієнта, відомі як Google, так і вашій програмі.

Набір значень залежить від типу програми, яку ви створюєте. Наприклад, програма JavaScript не вимагає секрету, але програма веб-сервера вимагає.

2. Отримайте маркер доступу на сервері авторизації Google. Перш ніж ваша програма зможе отримати доступ до приватних даних за допомогою Google API, вона повинна отримати маркер доступу, який надає доступ до цього API. Один маркер доступу може надавати різний ступінь доступу до декількох API. Змінний параметр, що називається область, контролює набір ресурсів та операцій, дозволених маркером доступу. Під час запиту на маркер доступу ваш додаток надсилає одне або кілька значень у параметрі сфери. Існує кілька способів зробити цей запит, і вони залежать від типу програми, яку ви створюєте. Наприклад, програма JavaScript може запитувати маркер доступу за допомогою перенаправлення браузера на Google, тоді як програма, встановлена на пристрої, який не має браузера, використовує запити веб-сервісу. Деякі запити вимагають кроку автентифікації, коли користувач входить у свій обліковий запис Google. Після входу в систему користувача запитують, чи готовий він надати один або кілька дозволів, які вимагає ваша програма. Цей процес називається згодою користувача. Якщо користувач надає принаймні один дозвіл, сервер авторизації Google надсилає вашій програмі маркер доступу (або код авторизації, який ваша програма може використовувати для отримання маркера доступу) та список обсягів доступу, наданих цим маркером. Якщо користувач не надає дозволу, сервер видає помилку. Як правило, найкращою практикою є запит на обсяги поступово, у той час, коли потрібен доступ, а не наперед. Наприклад, програма, яка хоче підтримати збереження події в календарі, не повинна запитувати доступ до Календаря Google, поки користувач не натисне кнопку "Додати до календаря"; див. Додаткова авторизація.

3. Вивчіть обсяги доступу, надані користувачем. Порівняйте сфери дії, що входять у відповідь маркера доступу, із сферами, необхідними для доступу до функцій та функціональних можливостей вашої програми, залежно від доступу до відповідного API Google. Вимкніть будь-які функції вашого

додатка, які не можуть працювати без доступу до відповідного API. Обсяг, включений у ваш запит, може не збігатися з обсягом, включеним у вашу відповідь, навіть якщо користувач надав усі запитувані області. Зверніться до документації до кожного API Google, щоб дізнатись про обсяги, необхідні для доступу. API може зіставити кілька значень рядка області дії в одну область доступу, повертаючи однаковий рядок області для всіх значень, дозволених у запиті.

4. Надішліть маркер доступу API. Після того, як програма отримує маркер доступу, вона надсилає маркер Google API у заголовку запиту авторизації HTTP. Можна надсилати маркери як параметри рядка запиту URI, але ми не рекомендуємо цього, оскільки параметри URI можуть потрапити до файлів журналів, які не є повністю захищеними. Крім того, хороша практика REST уникати створення непотрібних імен параметрів URI. Зверніть увагу, що підтримка рядка запитів буде припинена 1 червня 2021 року. Маркери доступу дійсні лише для набору операцій та ресурсів, описаних у обсязі запиту маркера. Наприклад, якщо для API Google Calendar видано маркер доступу, він не надає доступу до API Google Contacts. Однак ви можете кілька разів надсилати цей маркер доступу до API Календаря Google для подібних операцій.

5. За потреби оновіть маркер доступу. Токени доступу мають обмежений термін служби. Якщо вашій програмі потрібен доступ до API Google після закінчення терміну дії одного маркера доступу, він може отримати маркер оновлення. Маркер оновлення дозволяє вашій програмі отримувати нові маркери доступу.

Послідовність авторизації починається, коли ваша програма переспрямовує браузер на URL-адресу Google; URL-адреса включає параметри запиту, які вказують на тип доступу, що запитується. Google здійснює автентифікацію користувача, вибір сеансу та згоду користувача. Результатом є код авторизації, який програма може обміняти на маркер доступу та маркер оновлення.

Додаток повинен зберігати маркер оновлення для подальшого використання та використовувати маркер доступу для доступу до Google API.

Як тільки термін дії маркера доступу закінчується, програма використовує маркер оновлення для отримання нового.

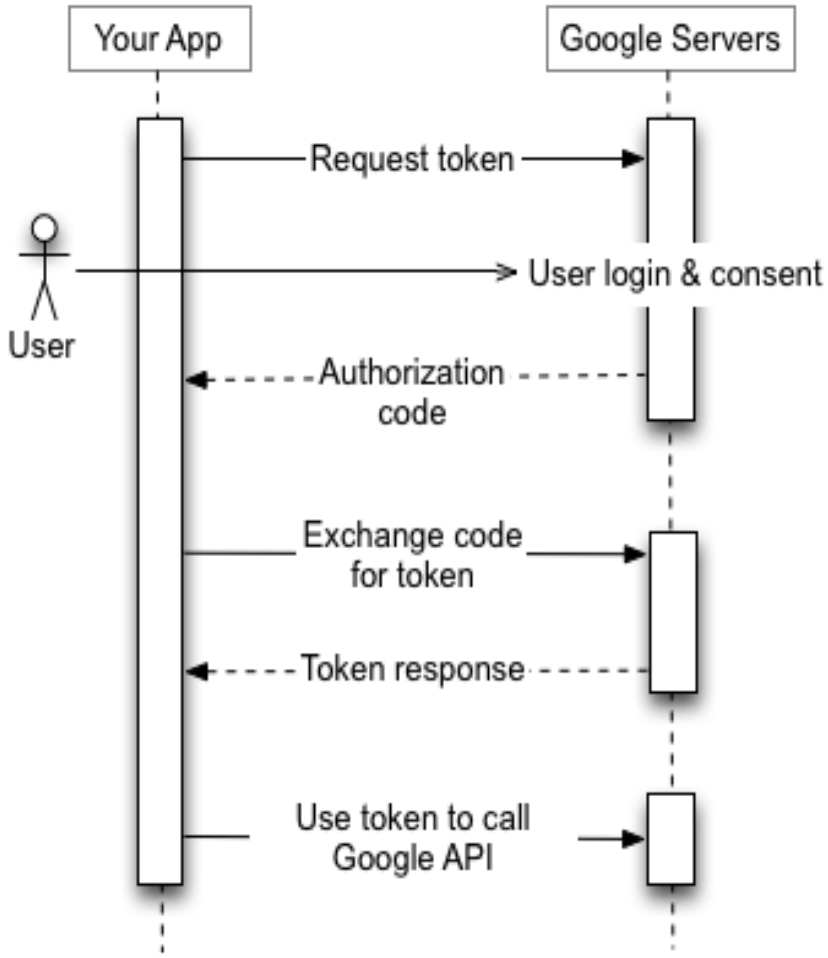

Рисунок 3.1 OAuth2 Flow.

Кінцева точка Google OAuth 2.0 підтримує програми, встановлені на таких пристроях, як комп'ютери, мобільні пристрої та планшети. Коли ви створюєте ідентифікатор клієнта через консоль Google API, вкажіть, що це встановлена програма, а потім виберіть тип програми Android, додаток Chrome, iOS, універсальна платформа Windows (UWP) або робочий стіл. Результатом процесу є ідентифікатор клієнта, а в деяких випадках і секрет клієнта, який ви

вставляєте у вихідний код своєї програми. (У цьому контексті секрет клієнта, очевидно, не трактується як секрет.) Послідовність авторизації починається, коли ваша програма переспрямовує браузер на URL-адресу Google; URL-адреса включає параметри запиту, які вказують на тип доступу, що запитується. Google здійснює автентифікацію користувача, вибір сеансу та згоду користувача. Результатом є код авторизації, який програма може обміняти на маркер доступу та маркер оновлення. Додаток повинен зберігати маркер оновлення для подальшого використання та використовувати маркер доступу для доступу до Google API. Як тільки термін дії маркера доступу закінчується, програма використовує маркер оновлення для отримання нового.

Клієнтські програми (JavaScript): Кінцева точка Google OAuth 2.0 підтримує програми JavaScript, які працюють у браузері. Послідовність авторизації починається, коли ваша програма переспрямовує браузер на URL-адресу Google; URL-адреса включає параметри запиту, які вказують на тип доступу, що запитується. Google здійснює автентифікацію користувача, вибір сеансу та згоду користувача. Результатом є маркер доступу, який клієнт повинен перевірити, перш ніж включити його до запиту Google API. Коли термін дії маркера закінчується, програма повторює процес.

Рисунок 3.2 OAuth2 Flow для клієнтських додатків.

3.4 Використання системи

3.4.1 Системні вимоги

Додаток створений для веб-браузера, тому він буде підтримуватись у всіх сучасних браузерах таких як Chrome, Firefox і так далі.

Також в додатку підтримуєтсья мобільне розширення і це означає що ним можна такош користуватись з мобільних пристроів.

Додаток повинен мати у своїх кодах безпечне збереження даних, а також не може завантажувати, встановлювати і виконувати програмний код, який вносить, змінює функції чи функціональність інших програм.. Програмне забезпечення повинно виключати можливість передання вірусів, файлів чи шкідливого програмного коду, що може завдати шкоди опервційні системі так як через браузер немає можливості взаємодіяти з зовнішньою операційною системою. Оскільки додаток передбачає використання сповіщень, користувачі

мають бути ознайомлені з тим, що цей функціонал може скоротити час автономного використання пристроя.

3.4.2 Огляд та тестування системи

Бета-тестування - це найпопулярніза з методологій тестування, яка забезпечує розробникам, спосіб оцінити рівень задоволеності проектом кінцевими користувачами. Користувачі, які використовують продукт, в ході декого періоду часу провіряють програмне забезпечення на функціональність, надійність, зручність використання, сумісність.

У процесі та після проходження завершальної стадії тестування береться досвід роботи з додатком, кінцевих користувачів, які пропонують відгуки покращення функціоналу та інтерфейсу. Це все дуже допомагає оцінити якість та надійність кінцевого продукту.

Розглянемо два типи бета-тестування:

- закрите - програма тестується в обмеженій групі користувачів, які мають запрошення або спеціальний доступ.
- відкритте - дає можливість спробувати додаток великій кількості реальних користувачів та отримувати відповідні зворотні зв’язки. Будь-який користувач може залучитись до відкритого бета-тестування та надіслати свій розгорнутий відгук відгук.

Переваги такого тестування:

- Можливість оцінити проект з точки зору користувачів.
- Різноманітність людей які тестують продукт та мають різні очікування від нього, також можуть покривати різні сценарії користування.
- Можливість виявити проблеми на початкових стадіях які краще виправити до офіційного релізу.

Програмний додаток складається з чотирьох основних сторінок: сторінка авторизації, сторінка перегляду повідомлень, сторінка перегляду конкретного повідомлення, сторінка написання повідомлення. Перша сторінка яку користувач бачить при запуску додатку – сторінка авторизації. Вона являє собою кнопку з оформленням та подальшу форму вводу логіну та паролю. Ця сторінка може бути добре знайома тим хто користується почтою від Gmail. На ній ми маєм авторизуватись для того щоб отримати доступ до листів та іншої інформації. Форма буде складатись з вводу адреси електронної почти та паролю. Так як цю форму нам відправляє Google з міркувань підвищеної безкеки від взломів зловмисниками, ми не можемо змінити її дизайн і тому виглядає вона наступним чином:

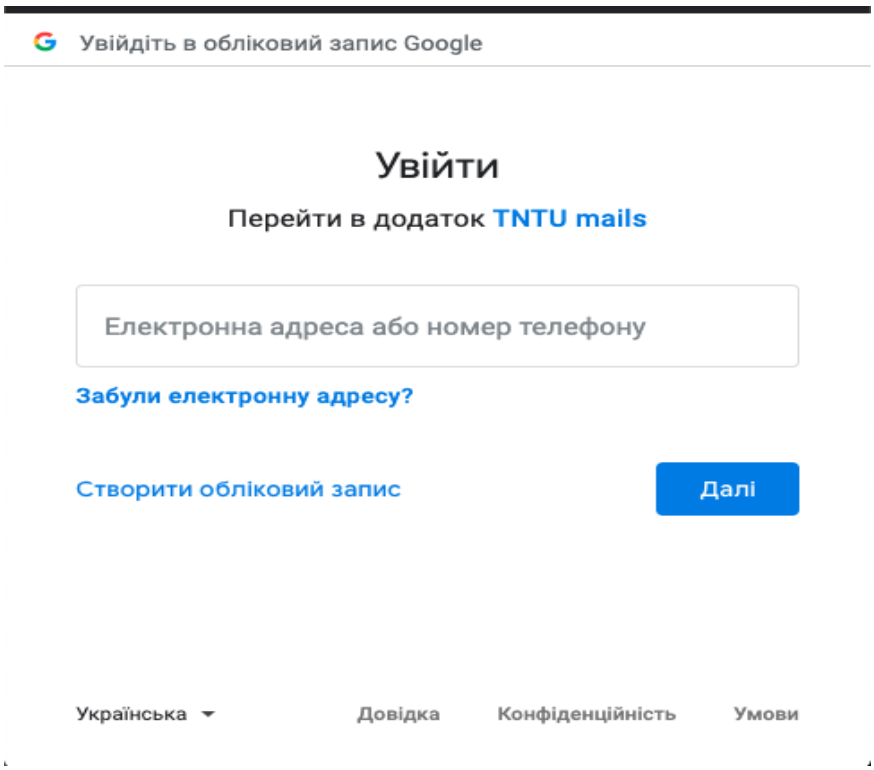

Рисунок 3.3 Сторінка авторизації

Наступна сторінка – це головна сторінка перегляду повідомлень, як можна побачити на ній є логотип університету, повідомлення, кнопка «Створити», та також зліва вкладки між якими можна переключатись на обрані повідомлення, та повідомленя які вже були надіслані. Для дизайну

використовувалась бібліотека компонентів Material UI, яку розробила Google тому по зовнішньому вигляду дизайн схожий на сервіси віж гугл.

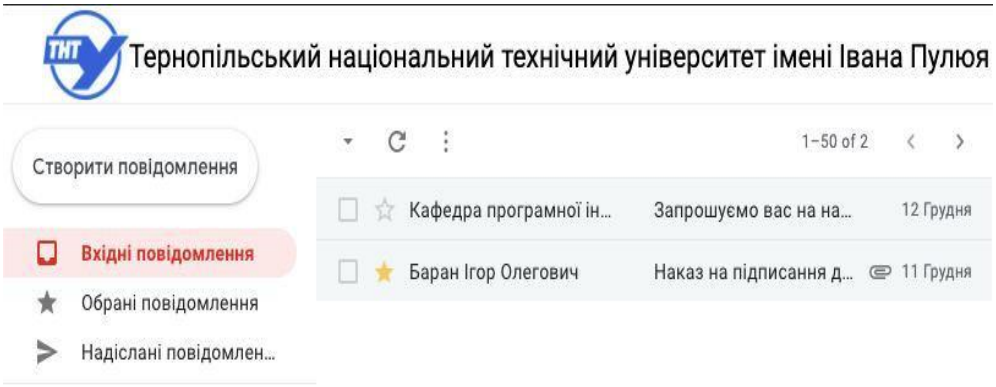

Рисунок 3.4 Перегляд повідомлень

Наступна сторінка – це сторінка перегляду конкретного повідомлеення. Після того як обрати повідомлення з загального списку і натиснути на нього ми будем бачити таку сторінку перегляду повідомлення, яка включає тему повідомлення, автора, дату, та сам текст повідомлення, тут можна натиснути на кнопку відповісти, яка перенаправить на сторінку надсилання повідомлення. Виглядає ця сторінка наступним чином:

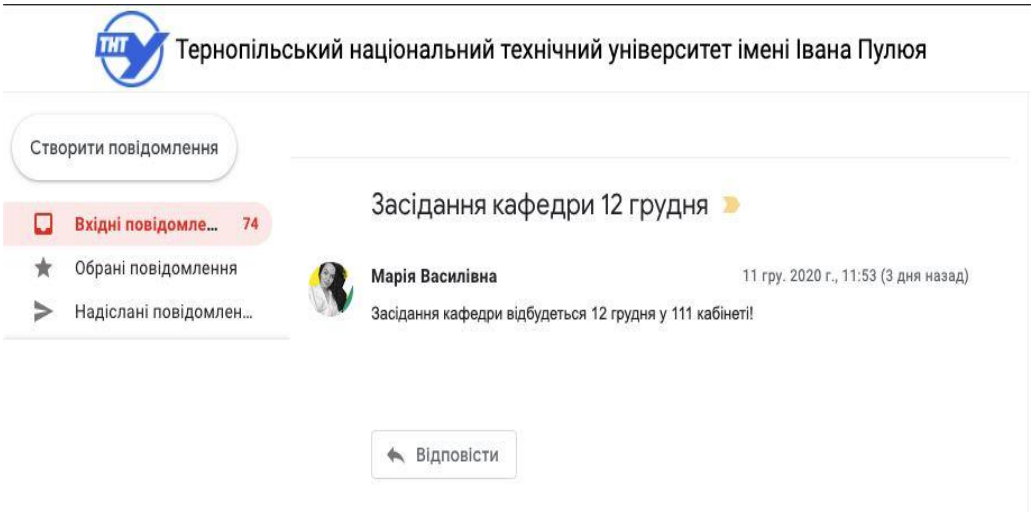

Рисунок 3.5 Перегляд конкретного повідомлення

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Створення програмного забезпечення повинно завжди відбуватись в умовах нанесення мінімальної шкоди здоров'ю тому хто працює над ним. Метою дипломної роботи магістра є розробка веб-клієнту для електронної почти університету, що означає виконання великого обсягу роботи за персональним комп'ютером протягом тривалого часу, тому тут є доцільним відзначити важливість дотримання усіх правил техніки безпеки та норм охорони праці під час роботи з електронно-обчислювальними машинами. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98. Даний документ описує: вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ, гігієнічні вимоги до параметрів виробничого середовища приміщень, гігієнічні вимоги до організації і обладнання робочих місць, вимоги до режимів праці і відпочинку, вимоги до профілактичних медичних оглядів. Значне зниження наслідків несприятливої дії на програмістів шкідливих та небезпечних факторів можна досягти за допомогою дотримання цих вимог.

Відповідно до встановлених гігієнічно-санітарних вимог (ГОСТ 12.1.005-88, СН 4088-86) роботодавець зобов'язаний забезпечити в приміщеннях з ВДТ оптимальні параметри виробничого середовища.

Природне освітлення в приміщеннях з ВДТ має здійснюватися через вікна, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні

екранів ПК і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Основні вимоги до виробничого приміщення для експлуатації ВДТ:

- воно не може бути розміщено у підвалах та цокольних поверхах;
- площа на одне робоче місце в такому приміщенні повинна становити не менше 6,0м2, а об’єм не менше 20,0 м3;
- воно повинно мати природне та штучне освітлення відповідно до ДБН В.2.5-28:2018;
- в ньому мають бути шафи для зберігання документів, магнітних дисків, полиці, стелажі, тумби тощо, з урахуванням вимог до площі приміщення;
- щоденно проводити вологе прибирання;
- поруч з приміщенням для роботи з ВДТ мають бути обладнані:
- побутова кімната для відпочинку під час роботи;
- кімната психологічного розвантаження.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним ВДТ, має здійснюватись системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні лампи ЛБ.

Вимоги до освітлення приміщень та робочих місць під час роботи з ВДТ:

- освітленість на робочому місці повинна відповідати характеру зорової роботи, який визначається трьома параметрами: об’єктом розрізнення – найменшим розміром об’єкта, що розглядається на моніторі ПК; фоном, який характеризується коефіцієнтом відбиття; контрастом об’єкта і фону;
- необхідно забезпечити достатньо рівномірне розподілення яскравості на робочій поверхні монітора, а також в межах навколишнього простору;
- на робочій поверхні повинні бути відсутні різкі тіні;

- в полі зору не повинно бути відблисків (підвищеної яскравості поверхонь, які світяться та викликають осліплення);
- величина освітленості повинна бути постійною під час роботи;
- слід обирати оптимальну спрямованість світлового потоку і необхідний склад світла. Застосування світильників без розсіювачів та екрануючих гратів заборонено.

Гігієнічні норми до організації і обладнання робочих місць з ВДТ. При розташуванні елементів робочого місця користувача ВДТ слід враховувати:

- робочу позу користувача;
- простір для розміщення користувача;
- можливість огляду елементів робочого місця;
- можливість ведення захистів;
- розміщення документації і матеріалів, які використовуються користувачем.

Конструкція робочого місця користувача ВДТ має забезпечити підтримання оптимальної робочої пози. Робочі місця з ВДТ слід так розташувати відносно вікон, щоб природне світло падало збоку, переважно зліва.

Робочі місця з ВДТ повинні бути розташовані від стіни з вікнами на відстані не менше 1,5м, від інших стін — на відстані 1 м, відстань між собою – не менше ніж 1,5 м.

Принтер повинен бути розміщений у зручному для користувача положенні, так, що максимальна відстань від користувача до клавіш управління принтером не перевищувала довжину витягнутої руки користувача.

Конструкція робочого стола повинна забезпечувати можливість оптимального роміщення на робочій поверхні обладнання, що використовується, з врахуванням його кількості та конструктивних

особливостей (розмір монітора, клавіатури, принтера, ПК та ін.) і документів, а також враховувати характер роботи, що виконується.

Вимоги до режимів праці і відпочинку при роботі з ВДТ. Під час роботи з ВДТ для збереження здоров'я працівників, запобігання профзахворюванням і підтримки працездатності встановлюються внутрішньо змінні регламентовані перерви для відпочинку.

Тривалість регламентованих перерв під час роботи з ЕОМ за 8-годинної денної робочої зміни залежно від характеру праці: 15 хвилин через кожну годину роботи – для розробників програм зі застосуванням ЕОМ; 15 хвилин через кожні дві години – операторів із застосуванням ЕОМ; 10 хвилин після кожної години роботи за ВДТ для операторів комп'ютерного набору.

У випадках, коли виробничі обставини не дозволяють стосовувати регламентовані перерви, тривалість безперервної роботи з ВДТ не повинна перевищувати 4 годин.

Для зниження нервово-емоційного напруження, втомленості зорового аналізатора, для поліпшення мозкового кровообігу і запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які передбачені ДСанПіН 3.3.2.007-98, в тому числі і для сеансів психологічного розвантаження у кімнаті з відповідним інтер'єром та кольоровим оформленням.

Виконання вимог ДСанПіН 3.3.2.007-98 повинне стати нормою всіх користувачів, які працюють над даним проектом.

4.2 Забезпечення безпеки життєдіяльності при роботі з ПК

Робоче місце — це зона простору, що оснащена необхідним устаткуванням, де відбувається трудова діяльність одного працівника чи групи працівників.

Раціональне планування робочого місця має забезпечувати: найкраще розміщення знарядь і предметів праці, не допускати загального дискомфорту, зменшувати втомлюваність працівника, підвищувати його продуктивність праці. Площа робочого місця має бути такою, щоб працівник не робив зайвих рухів і не відчував незручності під час виконання роботи. Важливо мати також можливість змінити робочу позу, тобто положення корпусу, рук, ніг. Проте доцільно виключати або мінімізувати всі фізіологічно неприродні і незручні положення тіла.

Проведені дослідження показують, що при раціональній організації робочих місць продуктивність праці зростає знати на 15-25%.

Антропометричні вимоги визначають відповідність конструкцій техніки антропометричним характеристикам людини (зріст, розміри тіла та окремі рухові ланки). Показниками е раціональна робоча поза, оптимальні зони досягнення, раціональні трудові рухи.

Фізіологічні та психофізіологічні вимоги визначають відповідність техніки і середовища можливостям працівника щодо сприйняття, переробки інформації, прийняття і реалізації рішень.

Організація робочого місця передбачає:

- правильне розміщення робочого місця у виробничому приміщенні;
- вибір ергономічно обґрунтованого робочого положення, виробничих меблів з урахуванням антропометричних характеристик людини;
- раціональне компонування обладнання на робочих місцях;
- урахування характеру та особливостей трудової діяльності.
- Загальні принципи організації робочого місця:

- на робочому місці не повинно бути нічого зайвого. Усі необхідні для роботи предмети мають бути поряд із працівником, але не заважати йому;
- ті предмети, якими користуються частіше, розташовуються ближче, ніж ті предмети, якими користуються рідше;
- предмети, які беруть лівою рукою, повинні бути зліва, а ті предмети, які беруть правою рукою — справа;
- якщо використовують обидві руки, то місце розташування пристосувань вибирається з урахуванням зручності захоплювання його двома руками;
- робоче місце не повинно бути захаращене;
- організація робочого місця повинна забезпечувати необхідну оглядовість.

Статичні напруження працівника в процесі праці пов'язані з підтриманням у нерухомому стані предметів і знарядь праці, а також підтриманням робочої пози.

Робоча поза — це основне положення працівника у просторі: зручна робоча поза має забезпечувати стійкість положення корпусу, ніг, рук, голови працівника під час роботи, мінімальні затрати енергії та максимальну результативність праці.

Найпоширенішими у процесі праці є пози сидячи і стоячи. Проектуючи робоче місце, потрібно враховувати, що при виконанні роботи з фізичним навантаженням бажана поза стоячи, а при малих зусиллях — сидячи.

Робоча поза стоячи втомлює людину більше, ніж сидяча. Вона вимагає на 10 % більше енергії, спричиняє підвищення артеріального і венозного тиску крові, розширення вен на ногах, пошкодження ступень, викривлення хребта.

Під час роботи сидячи нижня частина корпуса розслаблена, а основне статичне навантаження припадає на м'язи шиї, спини, таза, стегон. Неправильна сидяча

поза може викликати застій крові в ногах, а якщо виконується великий обсяг роботи для пальців рук — запалення суглобів.

4.2.1 Параметри робочого місця

Загальні ергономічні вимоги для організації робочого місця користувача ПЕОМ (ГОСТ 12.2.049-80, ГОСТ 122032-78, ГОСТ 22269-76). Ці вимоги встановлюють основні параметри робочого місця, оснащеного дисплеєм, і враховують особливість виконуваних робіт.

Площа кабінету, в якому буде проходити робота повинна бути не менш 6 м², а об’єм не менш 24 м³. Для внутрішньої обробки приміщення повинні використовуватися дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі – 0,7-0,8; для стін – 0,5-0,6; для підлоги – 0,3-0,5.

Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання. Конструкція крісла повинна забезпечувати підтримку раціональної робочої пози під час роботи з відео-дисплейним терміналом (Далі ВДТ) і ПЕОМ, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми працюючого (згідно з ГОСТ 12.2.032-78). Поверхня сидіння, спинки та інших елементів стільця (крісла) повинна бути напівм'якою, з покриттям, що не електризується, неслизьке та повітронепроникне, що забезпечує легке очищення від забруднення.

Висота робочої поверхні столу, за відсутності можливості її регулювання повинна складати 725 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною – не менше 500 мм, не менше 450 мм в глибину на рівні колін і на рівні простягнутої ноги – не менше 650 мм. Робоче місце має бути обладнане підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах 150 мм за кутом нахилу опорної поверхні підставки до 20 градусів.

Відстань від очей користувача до екрану дисплея має становити 500-700 мм. Кут зору 10-20°, але не більше 40°; кут між верхнім краєм дисплея і рівнем очей користувача має становити не менше 10°. Кращим є розташування екрану перпендикулярно до лінії зору користувача.

Робочі місця по відношенню до світлових прорізів повинні розташовуватися не ближче 3 м так, щоб природне світло падало збоку, переважно зліва. Освітленість також впливає на стан здоров'я і працездатність людини. У відповідності зі СНіП 11-4-79 встановлені наступні вимоги до освітленості:

Для штучного освітлення:

- Комбіноване освітлення – освітленість 1500 лк;
- Загальне освітлення – освітленість 400 лк.

Для природного освітлення:

- Верхнє або комбіноване освітлення – коефіцієнт природної освітленості (далі КПО) 10%;
- Бічне освітлення – КПО 3.5%.

Для суміщеного освітлення:

- Верхнє або комбіноване освітлення – КПО 3-6%;
- Бічне освітлення – КПО 1.1-2%.

До основних показників, що визначають умови здорової роботи, належать: фон, контраст об'єкта з фоном, видимість, показник осліпленості, коефіцієнт пульсації освітленості.

Фон характеризується коефіцієнтом відбиття. Контраст об'єкта з фоном (К) характеризується співвідношенням яскравості розглянутого об'єкта (точки, лінії, знаки) і фону. Оскільки роботи користувача ПЕОМ відносяться до категорії 1а – легкі фізичні роботи (роботи проводяться сидячи і супроводжуються незначним фізичним напруженням, з енерговитратами до 120 ккал / годину), необхідно дотримуватися наступних норм: коефіцієнт

відображення більше 0,4, тобто світлий фон; контраст об'єкта з фоном великий і середній при К більше 0,2 (згідно СНіП 11-4-79).

У полі зору користувача ПЕОМ має бути забезпечений відповідний розподіл яскравості. Відношення яскравості екрана до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1 (СНіП 11-4-79). У зв'язку з цим дисплей ПЕОМ повинен відповідати наступним вимогам:

- яскравість свічення екрану не менше 100 кд/м;
- мінімальний розмір світної точки для кольорового дисплея не більше 0,6 мм ;
- контрастність зображення знаку – не менше 0,8;
- низькочастотне тремтіння зображення в діапазоні 0,05-1,0 Гц повинно знаходитися в межах 0,1 мм;
- екран повинен мати покриття антивідблиску;
- відеомонітор повинен бути обладнаний поворотним майданчиком, що дозволяє переміщати відеотермінал в горизонтальній і вертикальній площинах в межах 130-220 мм і змінювати кут нахилу на 10-15 мм.

Коефіцієнт відбиття світла матеріалами і обладнанням всередині приміщень має велике значення для освітлення: чим більше світла відбивається від поверхонь, тим вище освітленість. Коефіцієнт відображення відповідно повинен бути для: стелі 60-70%, стін 40-50%, підлоги 30%, для інших поверхонь 30-40%.

Результати досліджень показують, що найбільшою мірою негативний фізіологічний вплив на операторів ПК пов'язаний з дискомфортними зоровими умовами через неправильно спроектоване освітлення. Згідно СНіП II-4-79 освітленість на горизонтальній площині робочого місця оператора ЕОМ повинна складати 400 лк при висоті цій площині 0,8 м над підлогою.

4.2.2 Вимоги до освітленості і повітряного середовища в робочій зоні

Світловий клімат визначає зоровий дискомфорт. Запобігти шкідливому впливу освітлення можна шляхом правильного підбору системи освітлення,

джерел світла (за їх спектрального складу випромінювання), світильників. Коли штучне світло змішується з природним, рекомендується використовувати лампи за спектральним складом найбільш близькі до сонячного світла. Світильники слід вибирати з розсіювачами, а блискучі деталі освітлювального обладнання, що можуть потрапити в поле зору оператора, повинні бути замінені на матові. Розташовувати робоче місце, обладнане дисплеєм, необхідно таким чином, щоб у полі зору оператора не потрапляли вікна або освітлювальні прилади; вони не повинні знаходиться і безпосередньо за спиною оператора. Вікна в приміщеннях з дисплеями обладнають шторами з коефіцієнтом відображення 0,5 ... 0,7, стіни фарбують матовою фарбою з коефіцієнтом відображення 0,4 ... 0,6. Світловий клімат може бути поліпшений шляхом встановлення спеціальних антивідблискових контрастних фільтрів, однак при виборі типу фільтра необхідно враховувати умови роботи з комп'ютером, оскільки оптимальні значення коефіцієнтів пропускання і дзеркального відображення фільтрів залежать від освітленості робочого місця і типу джерела світла.

Враховуючи великий вплив освітлення на працездатність оператора при роботі з комп'ютером, проведемо розрахунок необхідної освітленості в приміщенні з дисплеями при наступних умовах: гігієнічна норма освітленості на горизонтальній поверхні на рівні робочого місця оператора – 400 лк; ширина приміщення – 7 м, довжина – 8 м, висота – 3 м. Коефіцієнт відбиття від стелі – 70, від стін – 50, від робочих поверхонь – 30. Повітряне середовище – нормальне (вміст пилу, диму й кіптяви не більше 5 мг/м³).

Повітряне середовище в робочій зоні визначається мікрокліматом виробничого приміщення. Величини температури, відносної вологості та швидкості руху повітря на робочих місцях з дисплеями повинні відповідати допустимим значенням, які встановлені ГОСТ 12.1.005-88 ССБТ для категорії робіт 1а (легкі фізичні роботи, вироблені сидячи і супроводжуються незначною фізичною напругою до 120 ккал/год.). Згідно з цим документом допустимі значення температури повітря в приміщенні становлять 19-25 °С,

відносної вологості повітря – 55%, швидкості руху повітря на рівні особи – 0,1 м/С. При наявності досить комфортного робочого середовища атмосферний тиск по ГОСТ 21552-84 ССБТ може змінюватися від 84 до 107 кПа (630 ... 800 мм рт. ст.).

Шум несприятливий для людини, особливо при тривалому впливі. В оператора це виражається в зниженні працездатності (наприклад, швидкість обробки тексту зменшується на 10-15%), у прискоренні розвитку зорового стомлення, зміну відчуття кольору, підвищенні витрати енергії (на 17%). Тривалий та інтенсивний шум значно знижує продуктивність праці і призводить до зростання кількості помилок у роботі. У відділі головного економіста шум може створюватися телефонними дзвінками та розмовами, системними блоками та клавіатурою ПЕОМ. Так само джерелами шуму можуть бути системи кондиціювання та вентилювання повітря, існують і зовнішні джерела шуму (наприклад, працюють агрегати на вулиці).

4.2.3 Допустимі рівні звуку на робочих місцях

Допустимі рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам «Санітарних норм допустимих рівнів шуму на робочих місцях» № 3223-85. Згідно з цими нормами в приміщенні, де працює користувач ПЕОМ для забезпечення оптимальної робочої середовища рівень шуму не повинен перевищувати 60 дБ. Основними заходами боротьби з шумом згідно з ГОСТ 12.1.029-80 ССБТ є ліквідація або ослаблення джерела шуму шляхом застосування звукопоглинаючих матеріалів у конструкціях механізмів, використання коштів звукопоглинання і раціональна планування виробничого приміщення.

Випромінювання ПК можуть бути небезпечними для здоров'я. Низькочастотні поля при тривалому опроміненні сидять біля ПК людей можуть привести до порушень самих різних фізіологічних процесів. Відповідно до ГОСТ 27016-86 і ГОСТ 27954-88 потужність дози рентгенівського випромінювання у будь-якій точці простору на відстані 5 см

від екрану відеомонітора при 41 годинному робочому тижні не повинна перевищувати 100 мкР/год (0,03 мкР/с), а інтенсивність ультрафіолетового випромінювання – 10 Вт/м².

В даний час випускаються вибухобезпечні відеомонітори. За способом захисту людини від ураження електричним струмом дисплеї виготовляються відповідно з 1-м класом захисту за ГОСТ 25861-84, тому кабель живлення дисплея має вилку з трьома виводами, один з яких заземлюючий.

Для забезпечення ПДУ чинників робочого середовища на робочих місцях у необхідних випадках використовуються спеціальні засоби захисту працюючих. Способи захисту бувають активними і пасивними. Способи активного захисту засновані на виявленні джерел несприятливих факторів і вплив на них. У випадках неможливості здійснення активного захисту застосовується пасивна, за якої джерела несприятливих факторів залишаються, але здійснюються заходи, спрямовані на попередження вплив цих факторів на людину. Пасивна захист може бути колективного та індивідуального. Розглянемо колективні засоби захисту оператора ПК.

Висока температура повітря негативно позначається на функціональному стані людини. Всі основні електронні блоки ПК мають вбудовані вентилятори для забезпечення стабільних температурних режимів їх функціонування, тому при створенні комфортних умов роботи особливу увагу необхідно приділити шляхам відводу повітря (припливно-витяжної вентиляції). Для захисту від електростатичного потенціалу можуть бути використані згадані вище антиблискові контрастуючі фільтри на екрани дисплеїв.

ВИСНОВКИ

Результатом цієї магістерської роботи є отримання веб сайту, а точніше веб-клієнту для електронної почти tntu.edu.ua.

У ході виконання роботи було досліджено та розглянуто різні протоколи для відправки повідомлень, а також інтеграцію з Google через протокол OAuth2.

Перевагами цього програмного проекту є система перегляду повідомлень для електронної пошти університету з авторизацією через сервіси Google. Додаток був розроблений з використанням найновіших технологій розробки веб сторінок та програмного забезпечення, також плюсом є те що цей додаток підтримується мобільними пристроями, тому це дозволить швидке та легке впровадження подальшого використання з змогою розширення функціоналу проекту. На ранішньому етапі побудови та проектування системи була проаналізована предметна область продукту, виявлені та поставлені подальші завдання, визначено архітектуру продукту, описано різні варіанти використання та саму реалізацію програми. На самому завершальному етапі програмного продукту, було виявлено та описано самі основні та другорядні вимоги до користування розробленого програмного забезпечення, також було виконано тестування кінцевої розробки, тобто веб-сайту.

Цей програмний продукт є досить вдали, тому що він чудово виконує усі необхідні та поставлені завдання, працює достатньо швидко оскільки була використана технологія SPA та працює без помилок. Сайт має зручний та дуже зрозумілий інтерфейс який дозволяє легко навігувати по ньому та користуватись.

ПЕРЕЛІК ПОСИЛАНЬ

1. M. Chapple – OAuth2 a big pictire [Електронний ресурс]: Lifewire Tech Untagled – Режим доступу: https://www.lifewire.com/oauth2-1019735. – Останнє відвідування: 2020 – Назва з домашньої сторінки сайту.

2. Samual Sam – SPA Desing and Architecture[Електронний ресурс]: Turtorials Points Simply Easy Learning – Режим доступу: https://www.tutorialspoint.com/TypesSPA – Останнє відвідування: 2020 – Назва з домашньої сторінки сайту.

3. Justin Ellingwood – Web Applicaton Architecture: [Електронний ресурс]: Prisma – Режим доступу: https://www.prisma.io/blog/wep-app-architecture-1iz9u29nwn37 – Останнє відвідування: 2020 – Назва з домашньої сторінки сайту.

4. UML - Class Diagram [Електронний ресурс]: Turtorials Points Simply Easy Learning – Режим доступу: https://www.tutorialspoint.com/uml/uml_class_diagram.htm – Останнє відвідування: 2019 – Назва з домашньої сторінки сайту.

5. Shikhar Goel - JavaScript Programming Language [Електронний ресурс]: GeeksforGeeks | A computer science portal for geeks – Режим доступу: https://www.geeksforgeeks.org/js-programming-language/ – Останнє відвідування: 2019 – Назва з домашньої сторінки сайту.

6. Adam Cody - The Pros and Cons of 8 Popular Databases [Електронний ресурс]: KeyCDN - Content delivery made easy – Режим доступу: https://www.keycdn.com/blog/popular-databases#5-mongodb – Останнє відвідування: 2019 – Назва з домашньої сторінки сайту.

7. Системні вимоги [Електронний ресурс]: Вікіпедія - Системні вимоги – Режим доступу: https://uk.wikipedia.org/wiki/Системні_вимоги – Останнє відвідування: 2019 – Назва з домашньої сторінки сайту.

8. System requirements [Електронний ресурс]: Wikipedia - System requirements – Режим доступу:

https://en.wikipedia.org/wiki/System_requirements – Останнє відвідування: 2020 – Назва з домашньої сторінки сайту.

9. ДЖ.Грофф. React: Полное руководство: Пер. с англ. – 2-е изд.,перераб. И доп. / ДЖ.Грофф, П.Вайнберг; пер. с англ. под ред. В.Р. Гинзбурга. – К.:Издетельская группа BHV, 2001. – 816 с. –ISBN 5-7315-0102-5

10. Денисов В.В., Денісова І.А., Гутен В.В., Монтвіла О.І. Безпека життєдіяльності. Захист населення і територій при надзвичайних ситуаціях: Навч. посібник. - Москва: ІКЦ «МарТ», Ростов н / Д: Видавничий центр «МарТ», 2003. - 608 с.

11. Жидецький В. Ц. Охорона праці користувачів комп'ютерів. – Львів: Афіша, 2000. - 176 с.

12. Наказ Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду «Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин» від 26.03.2010 № 65 – Режим доступу: URL: http://zakon2.rada.gov.ua/laws/show/z0293-10

13. Правила безпечної експлуатації електроустановок споживачів [Текст] : ДНАОП 0.00-1.21-98. - Київ : Держнаглядохоронпраці, 2003. - 383 с.

14. Марков В.В. Основа здорового способу життя профілактика хвороб: навч. посібник для студ. вищ. пед. навч. закладів. - М.: Академія, 2001. - 320 с.

15. Ковальчук П.І. Моделювання і прогнозування стану навколишнього середовища. – Навч. посібник. — К.: Либідь, 2003 — 208 с.

16. Human Interface Guidelines [Електронний ресурс]: Developer - design – Режим доступу: https://developer.apple.com/design/human-interface-guidelines/ – Останнє відвідування: 2020 – Назва з домашньої сторінки сайту.

17. Тарасова В.В. Екологічна статистика – Навчальний посібник. – Київ. Центр учбової літератури, 2008. – 392 с.

18. Шматько В. Г., Нікітін Ю. В. Екологія і організація природоохоронної діяльності: навч. посіб. – 2-ге вид. – К.: КНТ, 2008. – 304 с.

ДОДАТКИ

ДОДАТОК А

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ІВАНА ПУЛЮЯ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку проекту

«Веб клієнт електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua»

Розробник:

виконавець ст. гр. СПм-51

Дрозд О.П.

_____

(підпис)

керівник проекту

Михалик Дмитро Михайлович

_____

(підпис)

Тернопіль 2020

ЗМІСТ

1 ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану на 2020 рік, та згідно наказу на виконання дипломної роботи студента-магістра.

Тема проекту: «Веб клієнт електронної пошти, що буде підтримуватись мобільними пристроями для пошти tntu.edu.ua».

2 ПРИЗНАЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Дипломна робота присвячена створенню веб-сайту для користування поштою tntu.edu.ua.

Предметом дослідження: є інтеграція з почтовими сервісами Google та використання Gmail API.

Мета роботи: розробка веб-сайту для спрощення доступу до пошти університету з підтримкою мобільних пристроїв.

За результатами виконаної роботи необхідно отримати веб-сайт з якого можна користуватись поштою tntu.edu.ua.

3 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Функціональні вимоги

Система повинна передбачати одну роль:

- користувач

Програмне забезпечення має виконувати наступні дії:

– авторизовуватись у поштових сервісах;

– відображати повідомлення;
– сортувати повідомлення;
– додавання повідомлень в обрані;
– відправку повідомлення;

3.2 Технічні вимоги

Данна програмна система працює під будь-яким браузером так як це веб-сторінка яка запускається в браузері, вона має більшу привязку до браузера аніж до операційної системи.

3.3 Програмні вимоги

Програмний продукт повинен коректно функціонувати на мобільних пристроях та в браузерах. Розроблювана програмна система повинна бути пристосована для використання будь-яким користувачем. Розробку виконувати з використанням мови JavaScript, бібліотеки React.

4. ЕТАПИ розробки

В ходів реалізації роботи проект повинен пройти крізь наступні стадії розробки:

- аналіз предметної області;
- проектування архітектури;
- реалізація програмних модулів;
- тестування результатів розробки;
- оформлення супровідної документації;
- здача роботи.

5. ПРОГРАМНА ДОКУМЕНТАЦІЯ

Для програмного продукту повинні бути розроблені наступні документи:

- Пояснювальна записка;

- Технічне завдання;

- Презентаційний матеріал;

- Інструкція користувача;

- Додатки.

Пояснювальна записка до проекту оформляється згідно діючих вимог до нормоконтролю проектів.

6. ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Розроблений програмний продукт має виконувати всі вимоги, що складаються з перерахованих у п. 3.1 характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначений у п.5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в п.1 цього документу.

7. ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ ТА ЗМІНИ В ПРОЕКТІ

Назва етапу	Відмітка*
Аналіз предметної області	
Архітектура системи	
Проектування веб-компонентів	
Використання системи	
Супровідна документація	

* відмітки про виконання етапу ставляться керівником проекту

ДОДАТОК Б

УДК 004.7

Технічні науки

ВЕБ-КЛІЄНТ ДЛЯ ЕЛЕКТРОННОЇ ПОЧТИ З ПІДТРИМКОЮ МОБІЛЬНИХ ПРИСТРОЇВ

Дрозд О.П., – ст.гр.СПм-61,
студент факультету інформаційних систем
Тернопільський національний технічний університет імені Івана Пулюя
м. Тернопіль, Україна

Дане дослідження присвячене розробці веб сайту для полегшення та покращення комунікації в межах університету та усунення проблеми у використанні почти з мобільних пристроїв. В роботі використовуються сучасні сервіси та технології для розробки веб додатків.

Однією з таких технологій є «Single Page Application». Ця технологія дозволяє нам розробляти клієнтські веб застосунки які будуть незалежні від веб-сервера. Данна технологія має перевагу над іншими, а саме може працювати без перезавантаження сторінки на відміну від звичайних клієнтських веб додатків. Це відбувається за рахунок того, що «Single Page Application» має тільки одну сторінку, та підставляє у неї різний вміст який приходить з веб-сервера. Таким чином веб сайти з використанням цієї технології працюють швидше ніж звичайні.

Також важливим завданням було інтеграція веб-клієнту з мобільними пристроями, тому що статистика показує, що середньостатистична людина проводить більше часу за мобільним пристроєм ніж за комп’ютером.. Ними зручніше користуватись і до них є доступ завжди. А за потреби можна навіть користуватися сайтом за відсутності інтернет-з’єднання. Для вирішення цієї проблеми було використано можливості мови розмітки «HTML» та каскадної таблиці стилів «CSS» за допомогою яких веб сайт було адаптовано під мобільне розширення.

Поставлена задача, а саме розробка веб-клієнта для електронної пошти була вирішена за допомогою інтеграції додатку з сервісами які надає «Google», а саме «Gmail API». За допомогою цього сервісу можна авторизуватись в електронній пошті, обмінюватись та переглядати повідомлення. Для того щоб зробити перегляд та обмін листами безпечніший було вирішено використати механізм авторизації у сервісах «Google» який називається «OAuth 2». За допомогою нього ми не зберігаєм ніде логін та пароль до електронної пошти, а отримуємо лише одноразовий маркер для доступу до неї, який буде оновлюватись з кожним разом як заходити на веб сайт.

Ця розробка перш за все принесе користь людям які користуються більше мобільними пристроями, тому що дозволить легше отримати доступ до електронної пошти. Також за допомогою протоколу авторизації «OAuth 2» можна гарантувати безпеку даних, тому що сайт їх не зберігає і важливі листи або файли не зможуть бути скомпрометованими.

Література

1. Single Page Application vs multi-page application [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

2. Що таке адаптивний дизайн сайту ? [Електронний ресурс] – Режим доступу до ресурсу: https://redstone.media/адаптивний-дизайн

ДОДАТОК В