Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

Сайт для розпорядку робочого дня

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення»

Керівник курсової роботи:
к. ф.-м. н. Гречко А. В.

Виконав студент:
Романюк О.А.

Київ 2022

(підпис)

Завдання отримав _____

(підпис)

Календарний план виконання курсової роботи

Тема: Розробка веб-застосування для планування дня

Календарний план виконання роботи:

№ п/п	Назва етапу курсового проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	07.10.2021	
2.	Ознайомлення з предметною областю	10.11.2021	
3.	Огляд і аналіз аналогів	25.11.2021	
4.	Пошук технологій для розробки	01.12.2021	
5.	Аналіз обраних технологій	25.01.2022	
7.	Початок створення практичної частини	01.03.2022	
8.	Початок написання теоретичної частини	05.04.2022	
9.	Подання проміжної версії теоретичної частини	03.05.2022	
10.	Подання проміжної версії практичної частини	06.05.2022	
11.	Корегування теоретичної частини	08.05.20202	
12.	Остаточне завершення практичної та теоретичної частини	10.05.2022	
13.	Створення презентації	11.05.2022	
14.	Захист курсової роботи	07.06.2022	

Студенту Романюк О. А.

Керівник Гречко А. В.

“_____”_____

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ 6
1.1.Аналіз сучасного стану питання та обґрунтування теми 6
1.2.Огляд існуючих аналогів розробки 7
1.3.Постановка задачі 7
РОЗДІЛ 2 Теоретичні відомості 8
РОЗДІЛ 3 Опис реалізації програмного продукту 9
3.1.Аналіз технічного завдання 9
3.2. Обгрунтування алгоритму й структури програми 9
3.3. Обгрунтування вибору засобів розробки 10
3.4. Опис розробки програми 11
3.5. Створення об’єктів і розробка головної програми . **Помилка! Закладку не визначено.**
3.6. Тестування програми і результати її виконання 13
3.7. Інструкція користувача 14
Висновки 16
Список використаної літератури 17
Додатки 18

ВСТУП

Наш сайт , який має назву “TimeManager” , це нова платформа , яка створена для спрощення процесу планування та розуміння своїх можливостей протягом робочого дня. На сьогоднішній момент проблема керування часом встає все більш актуальною. Це пов’язано із зростанням темпу життя нового прогресивного молодого суспільства , яке часто намагається поєднувати надзвичайно велику кількість задач і тим самим це часто призводить до не рівномірного розподілу навантаження , зменшення продуктивності , а то й можливо зовсім упущення виконання певних завдань. Дана тема курсової роботи була обрана мною саме тому , що це неодноразово доторкалось і мене самого. Саме тому мені цікаво буде дослідити сучасний стан платформ для розпорядку дня , дізнатись їх переваги та недоліки. Як результат , хочеться створити ефективну і зрозумілу будь якому користувачу систему , яка не обтяжуватиме користувача складними заповненням таблиць на день , а надати можливість швидко і якісно оформити свій тайм-менеджмент. Як вже зазначалось , цільовою аудиторію такої платформи стануть здебільшого молоді люди , які мають інтенсивний темп життя і відчувають різку потребу у впорядкуванні їх розкладу.

Мета курсової роботи – зробити свій внесок у покращення і вдосконалення сучасних методик по оптимізації часу для виконання задач , проектів та інших різних календарних подій.

Завдання курсової роботи – створити програмний застосунок з зручним та а інтуїтивно-зрозумілим інтерфейсом, де користувачі зможуть легко і якісно керувати своїм часом

Об’єктом дослідження є сучасний стан тайм-менеджменту в Україні, наявність та функціонал веб-сервісів, спрямованих на вирішення поставлених задач у суспільстві.

Предметом дослідження є способи отримання якісних послуг по впорядкуванні свого розпорядку дня в Україні та можливостях їх покращення.

В результаті дослідження виявлено, що існує надзвичайно велика кількість веб-сайтів та інших платформ , на яких можна спробувати організувати свій робочий день. Однак наявність такої великої кількості застосунків вирішує лише проблему дружності інтерфейсу , але ніяк не питання самого змісту і функціоналу , який необхідних сучасному користувачу.

Основною мовою програмування при розробці платформи стала Java , використовуючи Spring Framework. Основна робота виконувалась у середовищі Intellij IDEA від JetBrains.

Всі дані зберігаються у PostgresSQL , для роботи з нею було використано програму з графічним інтерфейсом pgAdmin Розробка графічного інтерфейсу виконувалась з допомогою Thymeleaf.

Курсова робота складається з вступу та трьох основних розділів: "Аналіз предметної області. Постановка завдання курсової роботи", "Теоретичні відомості" та "Опис реалізації програмного продукту". У висновках підбиваються основні підсумки роботи. Посилання на наукові роботи та інші джерела інформації знаходяться у списку використаних джерел. Всі матеріали,

що доповнюють текст, вкладення та програмний код прикладної програми містяться у "Додатках".

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

1.1.Аналіз сучасного стану питання та обґрунтування теми

Надзвичайно поширена гіпотеза , що безлад може заважати нашому плануванню часу та продуктивності, адже чим більше завдань ми собі набираємо , чим більше намагаємось встигнути – тим важче довести хоча б одну із прав до логічного кінця. Зберігання організованості часто є важливим елементом правильного управління часом, оскільки це допомагає нам витрачати більше часу на досягнення цілей і менше часу на перемикання між завданнями або пошук предметів.

Сема тому у людей виникає необхідність вести список справ , щоб упорядкувати свої завдання . Зазвичай в нагоді приходяться блокноти або певні цифрові планувальники. Це одне з найпопулярніших рішень для вирішення проблеми тайм-менеджменту. Звичайно , зважаючи на рівень цифровізації суспільства все більш актуальним стає саме цифрові застосунки. Основне їх завдання на разі це візуалізувати усі події та завдання , встановити нагадування , щоб користувач мав можливість завчасно підготуватись до них.

Проблема якісної візуалізації , часто виноситься на перше місце у таких застосунках. Вони намагаються згрупувати і візуалізувати усі плани користувача і подати їх йому. Відповідно можна чітко побачити своє щоденне / тижневе навантаження , оптимізувати не зручні вікна та стати набагато продуктивнішим. Та на мою особисту думку це не вирішує багато більш важливих проблем , з якими зіштовхується користувач. Наприклад програма може показати гарно розписаний робочий день певного користувача з 12-00 по 15-00 , а наступний день навантаження може бути з 7-00 по 20-00. Зазвичай користувач мало що захоче змінювати і буде готуватись працювати за таким графіком. Це перш за все пов’язано із необізнаністю користувача в людській психології. Як стверджує психологиня Тамара Сухенко (джерело 1) , людський організм здатний активно працювати і споживати інформацію лише до 18-00 , а будь яка така діяльність опісля буде шкодити людину. Окрім того існує чимало порад від професіоналів , що слід коректно розподіляти навантаження в розрізі цілого тижня. Намагатися зробити розвантаженими такі дні як понеділок і середа , а решту днів навпаки напруженішими. Окрім того психологи радять брати більш складні і ємкі в часі завдання на ранок , коли ми продуктивніші. А більш прості і короткотриваліші на другу половину дня.

Актуальність цієї галузі підтверджується і великою кількістю різних тренерів і психологів які здатні допомогти людині організувати свої робочі процеси. Більшість з них дають надзвичайно важливі поради , але досі таки не існують програмних застосунків чи алгоритмів які здатні відповідно до цих рекомендації допомогти користувачу грамотно розподілити свій час.

Тож це підводить нас до актуальності створення нової платформи , в яку будуть вшиті корисні для користувача алгоритми , що будуть допомагати користувачу бути успішних і продуктивним.

1.2.Огляд існуючих аналогів розробки

В загальному доступі зараз доступно чимало програм , які мають певні переваги у використанні , але і недоліки. Зокрема схожу концепцію реалізовано у програмах : Todoist , Мої Завдання , Google Calendar тощо.

Давайте розглянемо деякі з них застосунок Todoist. Додаток , який має доволі простий зовнішній вигляд, реалізовує у собі такий функціонал :

- Створити подію , та назначити на неї час
- Фільтрація подій

Так , цей застосунок який має понад 10млн завантажень , має чимало недоліків. Це зокрема можливість реєстрації користувача , щоб можна було запрошувати чи повідомляти інших користувачів про ті чи інші події. Також доречно було б додати розділ подій на категорії : по пріоритетності виконанню , по часових межах тощо. Також не вражає спосіб візуалізації цих завдань.

Схожі недоліки присутні і у програмі “Мої справи” від українських розробників. Попри те , що у програмі реалізовано базову реєстрацію і логінізацію , виника чимало питань до самого функціонування запису і відображенні подій , через наявну монетизацію і наявність Pro-версії застосунку , яка є доволі не дешевою.

Здавалось , що такий гігант як Google мав би зважати на усі ці проблеми і спробувати вирішити їх. І справді , дана система має чудову візуалізацію. Система не обважена і у ній дуже легко можна зорієнтуватися. Але , Google Calendar та інші вище згадані застосунки не пробують аналізувати і шукати альтернативні часові вікна для виконання певних не великих задач. Ми можемо легко завантажити свій день і система не сповістить нас про перевантаження чи не оптимальність такого підходу до виконання задач. Саме це є на мою думку ключовою проблемою всіх наведених вище платформ.

1.3.Постановка задачі

В результаті проведеного аналізу , можна стверджувати , що дана предметна область є доволі проблематичною. Це перш за все пов’язано із швидким розвитком технологій , пришвидшення ритму життя серед молоді та певну застарілість і неактуальність уже існуючих рішень. Окрім того слід приділити увагу системі аналізуванні будь якого оремо взятого дня і тижня в цілому. Відповідно , одним із завдань , які я ставлю перед собою є реалізація унікальних алгоритмів аналізу робочого дня нашого користувача , з можливістю рекомендувати користувачу перерви або перенесення тих чи інших завдань в залежності від щоденного навантаження користувача. Вважаю , що такі алгоритми є цілком доцільними і вони зможуть допомогти нашому користувачу

якісно і продуктивно будувати свій робочий день. Звичайно слід приділити увагу і інтерфейсу. На перший план тут має вийти простота і зрозумілість для користувача

РОЗДІЛ 2 Теоретичні відомості

Чому чітка і конструктивна побудова графіка дня важлива? Насправді кожен з нас ввечері напередодні нового дня намагається уявляти те , чим він займатиметься завтра. Багато хто уявляє це доволі у схожому стилі : (прокинутись – зібратись – робота – обідня перерва – робота – добирання додому – хатні справи – вечеря- сон). Людина окреслює для себе лише загальні плани , не подаючись у деталі. Це може бути звичайний план умовного простого робочого , кожен день якого не відрізняється від попереднього. Та давайте уявимо гіпотетичного викладача , який працює в університеті. Щодня у нього по декілька пар, за різним розкладом. Окрім того постійні наради на кафедрі , чи зустрічі із колегами. Курси підвищення кваліфікації , можливо додаткова зайнятість, а ще наукова чи публікаційна робота , завдання які слід перевірити і тд. Відповідно це вносить надзвичайно багато деталей у такий загальний план як “робота”. Відповідно виникає потреба все це структурувати , адже мимохіть будучу чимось зайнятим і не встигаючи , можна навіть забути про щось. І чим більше таких речей трапляється , тим більше у людини накопичується боргів на наступний час.

Не слід забувати і про такий важливий чинник як вільний час , і те , що часто людина під час виконання якихось задач може відволікатись на соціальні мережі чи просто “залипнути” в телефоні на всякі дрібниці. Чому це важливо? Перш за все тому що , це дуже важливий фактор на те , що ми не встигатимемо виконати щось. Чіткий розклад який демонструватиме нам об’єм роботи буде відмежовувати у нас потребу відволікатись. Або ж , якщо цього не можна уникнути , то слід передбачити декілька вікон на відпочинок , які дозволять користувачу розслабитись. Але як би там не було , цей процес всерівно буде контрольованим і зменшить час який людина витратить марною.

Також слід визначити систему , по якій користувач зможе візуалізувати для себе інформацію. Очевидно , що користувач має бачити список усіх завдань у перспективі на день. Зазвичай прийнято демонструвати це у таблиці , яку можна інтерпретувати коротко як година-завдання. Це доволі наочно і зручно для користувача , оскільки у такій таблиці доволі легко зорієнтуватися. Також вважаю доцільно добавити такі інтерпретація у розрізі тижня , а також місяця. З власного досвіду користуваннями деякими застосунками , вважаю за доцільно добавити окремий список , де буде висвічуватись список усіх не виконаних завдань.

Тепер слід класифікувати усі події , які користувач може створювати. На мою думку доцільно було б розділяти їх на такі три категорії у відповідності до важливості тієї чи іншої події:

А) Надважливі

Б) Важливі
В) Рутинні(щоденні)

Я вирішив ділити на такі категорії, адже такий розподіл дасть змогу нам чітко відслідковувати і надавати перевагу тій чи іншій події. Але слід вирішити , як виділяти і розрізняти їх за необхідності. На мою думку найкраще вирішення , це застосування різних кольорів для зафарбування чи обведення тексту , яке різко вирізнить дану інформацію для користувача.

РОЗДІЛ 3 Опис реалізації програмного продукту

3.1.Аналіз технічного завдання

Основне призначення створюваного веб-застосунку – можливість планувати свій робочий день. Оскільки мета цього веб застосунку надавати послуги лише одній людині , було вирішено не розділяти функціонал на різні ролі , а зосередитись лише на користувачеві. Оскільки це особистий планувальник користувача , нам не потрібна взаємодія з іншими такими ж користувачами. Тому при реєстрації ,можна зосередитись лише на таких даних як логін (емейл) , пароль , прізвище та ім’я. Відповідно після того як користувач залогіниться на сайт , наш веб-застосунок має надавати йому наступний функціонал:

- Створити собі нову подію/нагадування/завдання
- Користувач може проходитись по застосунку і переглядати усі наявні справи у нього
- За необхідності користувач може видалити , або позначити як виконану певну подію.

3.2. Обгрунтування алгоритму й структури програми

Для реалізації веб-застосування було обрано Rest архітектуру. Це простий спосіб надсилання та отримання даних між клієнтом і сервером. Для передачі даних не потрібне програмне забезпечення чи стандарти. Він має попередньо визначену структуру для виклику API. Основною особливістю є те , що клієнт не має прямого доступу до даних. А все функціонування системи відбувається за допомогою запитів на сервер. Дана архітектура чудово вписується у контекст взаємодії нашого користувача із серверною частиною.

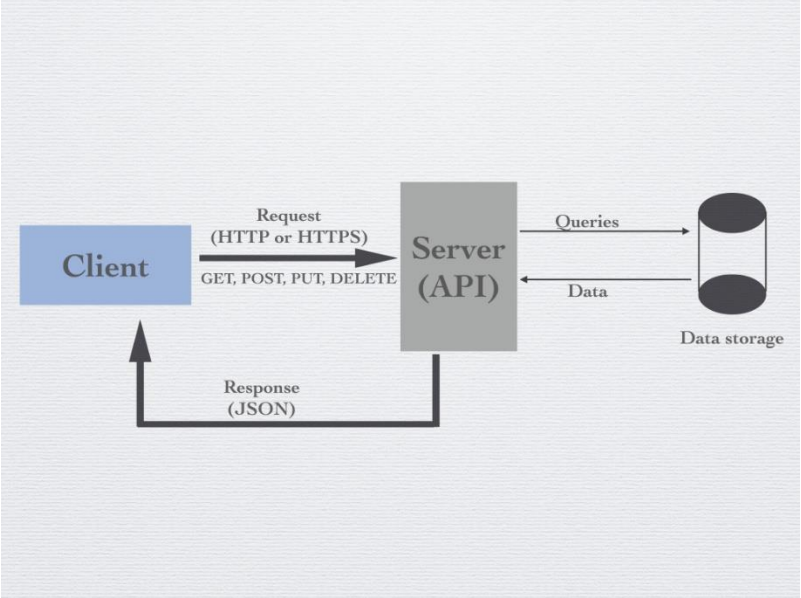


Рис. 2.1 Особливості архітектури Rest API

3.3. Обгрунтування вибору засобів розробки

Реалізація серверної(Backend) частини програми , було обрано мову програмування Java. Дана мова програмування , яка була створена із запозиченням синтаксису від C++ , вже більш ніж 25 років слугує інструментом для розробників ПЗ. Це пов’язано із тим , що дана мова програмування має чіткі переваги , такі як :

- Java є об’єктно орієнтована мова програмування. ООП є однією із найпоширеніших та найважливіших методологій розробки та ґрунтується вона на уявленні про програму як про сукупність об’єктів , саме це допомагає розробнику інтерпретувати людську повсякденність через програмний код.
- Java , є мовою високого рівня з доволі простим синтаксисом.
- Окрім того , мова програмування Java , має набір інструментів , які можуть позбавити від загальних вад безпеки ПЗ.

Для зберігання даних , мною було використано PostgreSQL. Дана база даних була обрана у зв’язку з тим , що вона добре інтегрується з мовою Java. Дана БД дозволяє визначити наші власні спеціалізовані функції. Структурована мова запитів Postgre має безліч функцій, які відсутні у її конкурентів.

3.4. Опис розробки програми

Розробка моєї програми почалась із творення репозиторію та ініціалізації його за допомогою Git. Дана система є надзвичайно корисною для розробників , оскільки дає змогу розробнику вертатись до старих напрацювань і тим самим ефективно віднаходити свої помилки. Після цього я створив публічний репозиторій на GitHub , де і розмістив свій репозиторій.

Наступним етапом стало створення структури власного проекту. На наступній картинці буде наведена загальна структуру проекту:

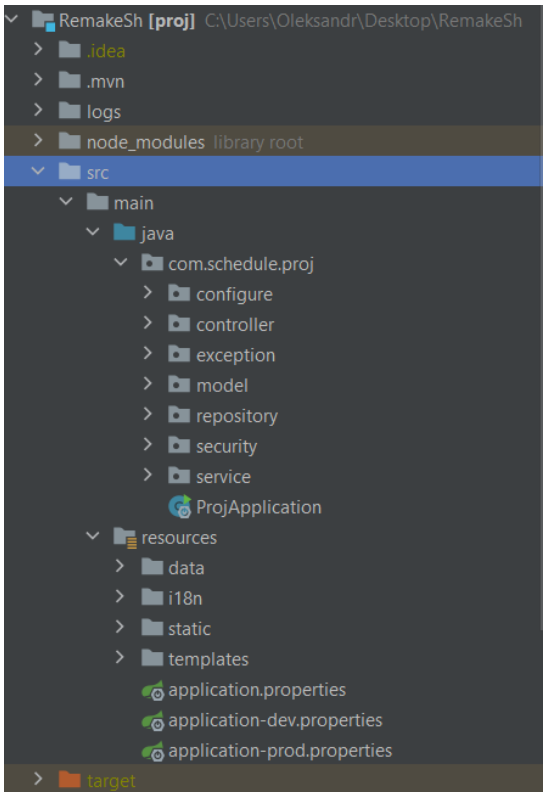


Рис. 2.2 Загальний каталог проекту

Також слід зауважити , що для обраної бази даних PostgresSQL , було створено відповідні налаштування і доданий докер файл з наступним кодом:

```
version: '3.8'
services:
  db:
    container_name: pg_container
    image: postgres
    restart: always
    environment:
      POSTGRES_USER: postgres
```

```
    POSTGRES_PASSWORD: root
    POSTGRES_DB: test_db
  ports:
    - "5432:5432"
pgadmin:
  container_name: pgadmin4_container
  image: dpage/pgadmin4
  restart: always
  environment:
    PGADMIN_DEFAULT_EMAIL: admin@admin.com
    PGADMIN_DEFAULT_PASSWORD: root
  ports:
    - "5050:80"
```

Наступним етапом стає створення репозиторію для фронтенду. Я вирішив добавити його у кореневий каталог нашого проекту , так як це доволі зручно коли ми використовуємо Thymeleaf. Також було завантажено деякі бібліотеки які буде використовувати фронтенд. На цьому етап початку розробки проекту можна вважати завершеним.

3.5 Опис файлів даних та інтерфейсу програми

На самому початку розробки програми , я вирішив виділити чітку структуру програми , де кожен пакет , буде відповідати за відповідно визначену йому функцію. Було створено:

- Package Model , для зберігання основних Entity
- Package Repository , яка відповідає за розміщення файлів необхідних для створення запитів і отримання необхідних даних
- Package Service , де розміщуватимуться класи , які реаізують ключову бізнес логіку гашу веб застосунку.
- Package Controller , де будуть реалізовані усіх ендпоінтів.
- Package Security , де реалізовується безпека нашого стосунку

Дана чітка архітектура розподілу класів може одразу ознайомити нас із тим , як саме функціонуватиме наша програма. Ключовим у наші програмі є реалізація Rest API , який допомагає клієнтським застосункам спілкуватися із серверами. Надзвичайно важливо правильно створити REST API, а також подбати про безпеку, продуктивність і простоту використання. Відповідно уся логіка необхідна для відпрацювання наших API буде відбуватися на Сервісах. Ті в свою

чергу можуть звертатись до бази даних за допомогою запитів до бази даних, чи звернень до наших Enteties. Також слід добавити , що наявні декілька DTO , які реалізують передачу даних між підсистемами програм.

3.6. Тестування програми і результати її виконання

Тестування веб-сайту проводилося у браузері Mozilla Firefox. Перш за все слід протестувати систему реєстрації і логінізації. В загальному вигляді , вона виглядає ось так :

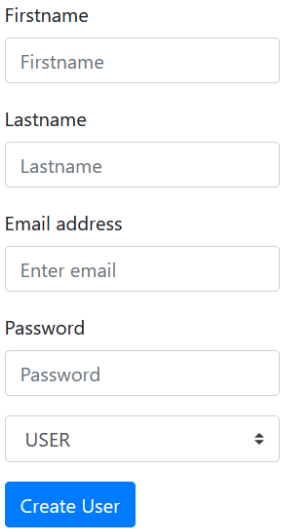


Рис. 2.3 Форма для реєстрації

При намаганні зареєструватись під ім’ям і прізвищем , яке менше 4 символів ми отримаємо зауваження від системи:

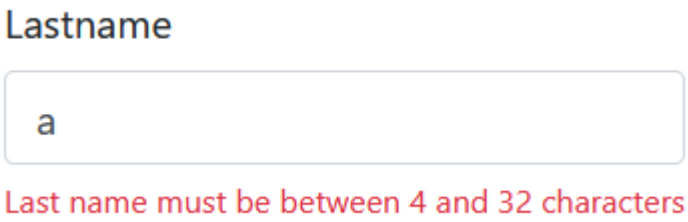


Рис. 2.4 Помилка введення невірних даних

Після введення коректних даних , ми перейдемо на сторінку логінації.

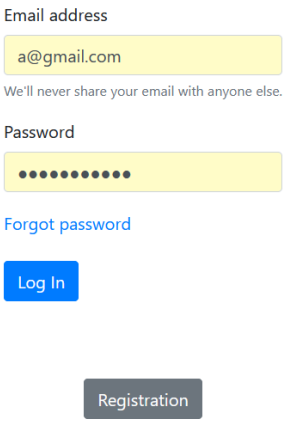


Рис. 2.5 Сторінка логінації

Також , система не пропустить пароль які не буде складатись з з букв , числа і великої букви.

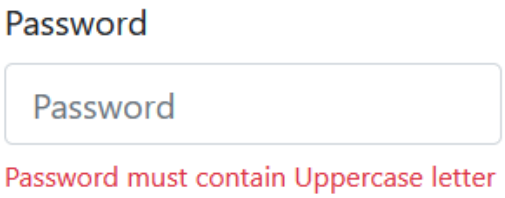


Рис. 2.6 Помилка при введені не правильного паролю

3.7. Інструкція користувача

Будь який користувач , який зайде на платформу для початку мусить зареєструватися. Слід зауважити , що користувач має ввечти ім’я і прізвище яке складається з більш ніж 4 символів. Електронна адреса має бути унікально. Якщо користувач уже нею користувався то він не зможе зайти під нею знову. У разі якщо користувач забув свій пароль під час логінації , він може скористуватися скиданням паролю. Це відбувається шляхом натискання на посилання :

Рис.3.1 Посилання на скидання паролю

У такому разі користувача переведе на нове вікно де він зможе ввести назву своєї електронної адреси. Якщо така пошта дійсно існує , користувач отримає

унікальний лист , де буде вказано новий пароль для входу в систему. Оскільки це рандомно-згенерований пароль , користувачу може бути не комфортно ним користуватися. Тоді після логінації , користувач може змінити пароль на той який він забажає.

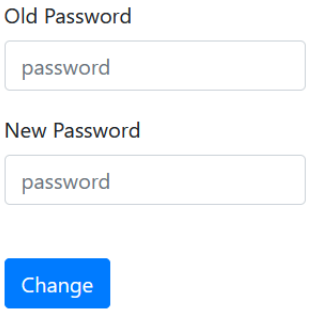


Рис.3.2 Сторінка оновлення паролю

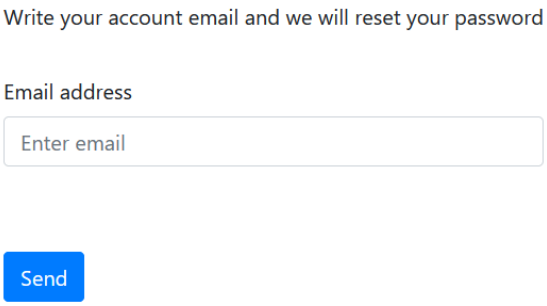


Рис.3.3 Сторінка скидання паролю

Коли користувач подолає усі труднощі , він може приступити до планування свого дня. Після простої форми додавання ,користувач може переглядати свої події у окремій вкладці. У додатку А наведено зображення , як користувач може бачити свій розклад у розрізі цілого місяці. Для кожної події , можна задати окремий колір , щоб це було більш інформативно. Також користувач може переглянути свій розклад у розрізі тижня , дня , або ж просто список всіх своїх планів. Усю дану інформацію наведено у додатках B-D.

Висновки

В ході написання курсової роботи було досліджено та проаналізовано вибрану
предметну область. Практичну частину було вирішено реалізувати у вигляді веб-застосунку.

Були проаналізовані сучасні додатки аналоги . У результаті цього , я виокремив для себе загальні тенденції , які є у цій сфері і спробував їх застосувати і для своєї платформи. Звичайно , намагаючись вирішити недоліки існуючих систем.

В результаті було створено новий застосунок , який здатний надавати користувачу весь необхідний функціонал для планування свого дня. Загалом система вийшла дружньою для користувача. Вона інтуїтивно зрозуміла , і не містить складних і не зрозумілих систем , в яких користувач міг би розгубитися.

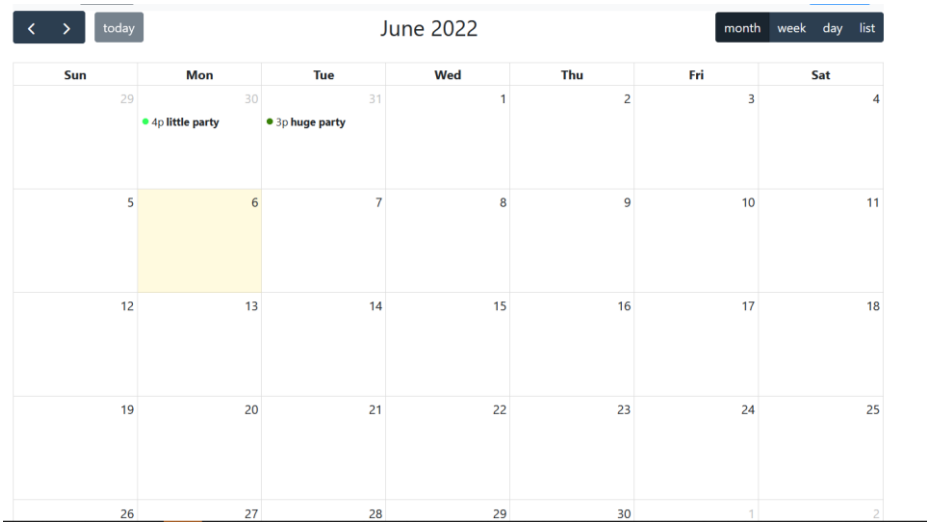
Даний застосунок є базовим , і у перспективі він може зазнати розширень і удосконалень. Це може бути додавання зв’язків між користувачами , комунікація між ними і можливість запрошувати на свої власні події. Також можна добавити систему оцінювання навантаження користувача і можливість кращого редагування публічних подій , які користувач запланував.

Список використаної літератури

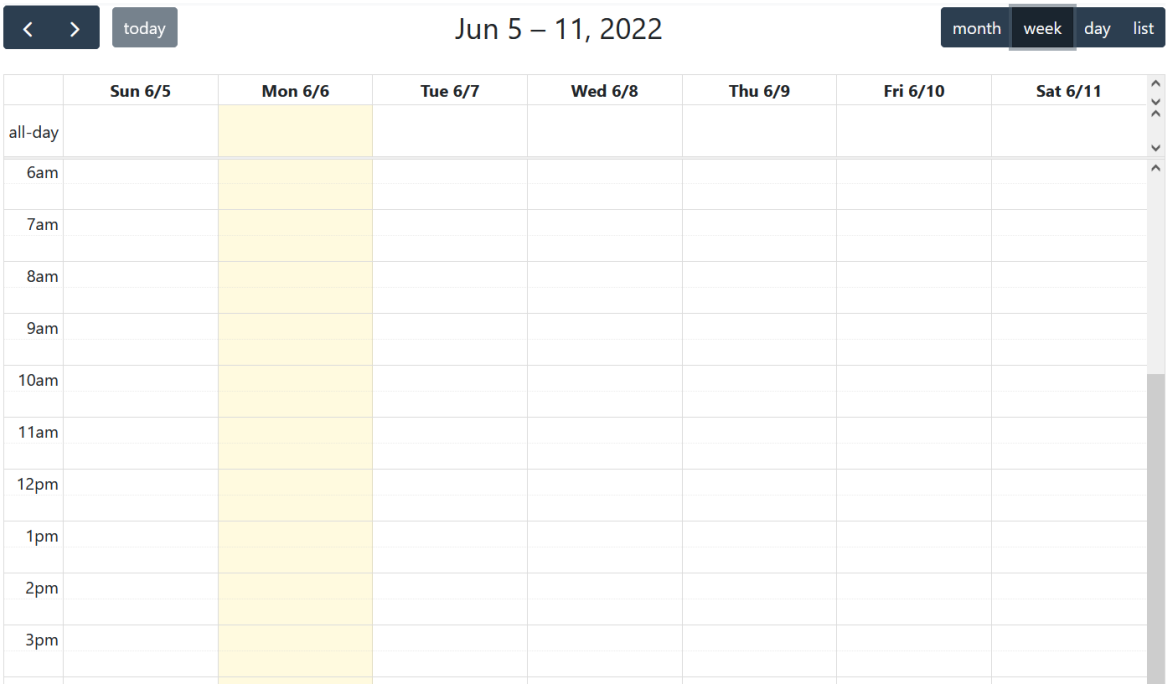
1. [Електронний ресурс] PostgreSQL database adapter for Java - https://www.psycopg.org/docs/
2. [Електронний ресурс] Documentation of FullCalendar - https://fullcalendar.io/
3. [Електронний ресурс] Ступ до мікросервісів - https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/
4. [Електронний ресурс] Spring Boot documentation https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
5. [Електронний ресурс] Spring Security - https://developer.mozilla.org/ru/docs/Web/Security
6. [Електронний ресурс] Як правильно формувати робочий день? - https://theukrainians.org/yak-splanuvaty-den/
7. [Електронний ресурс] Особливості людської психології у плануванні робочого дня - https://ifr.dcz.gov.ua/publikaciya/yak-vstygnuty-vse-za-robochyy-den
8. [Електронний ресурс] Посилання на мій гітхаб репозиторій , звідки я брав частину коду - https://github.com/OleksandRomaniuk

Додатки

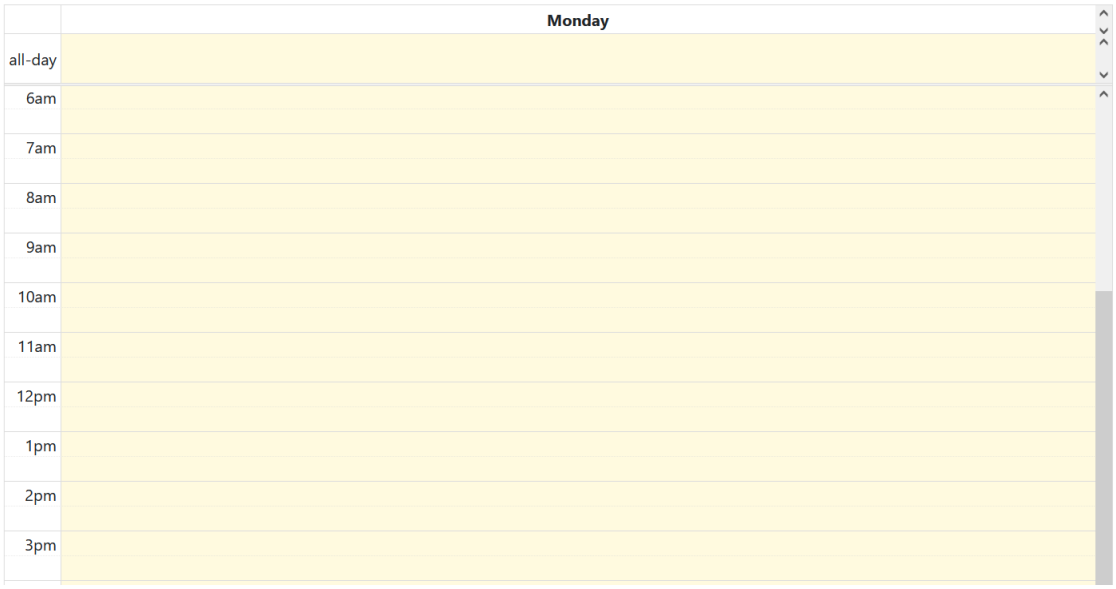
Додаток А



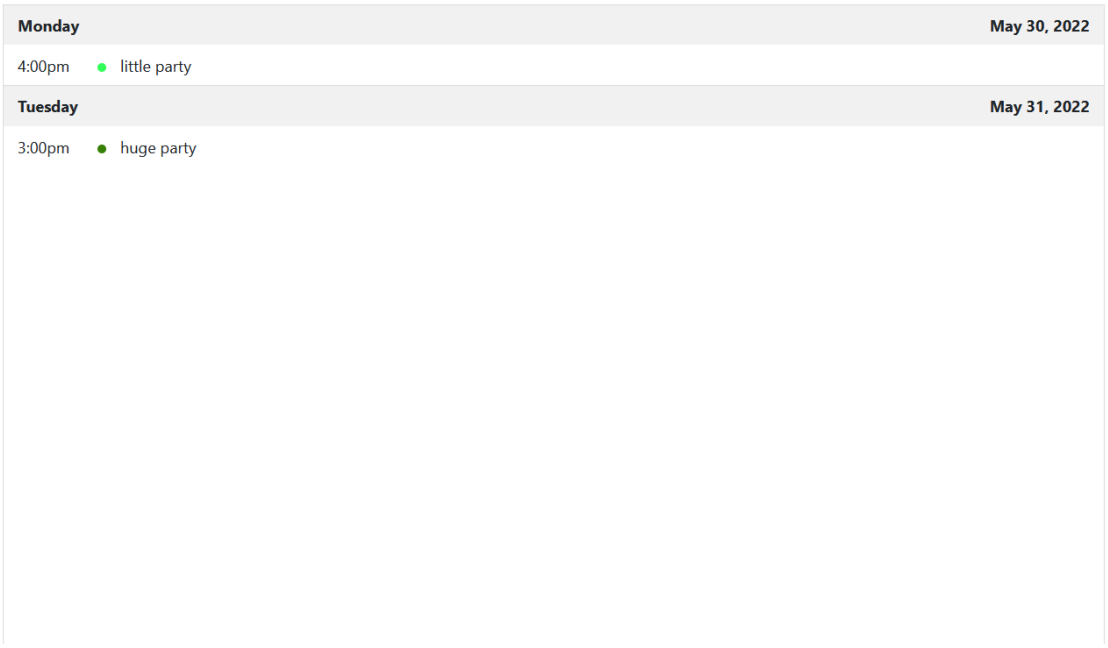
Додаток В



Додаток С


Monday
all-day
6am
7am
8am
9am
10am
11am
12pm
1pm
2pm
3pm

Додаток D


Monday
May 30, 2022
4:00pm
little party
Tuesday
May 31, 2022
3:00pm
huge party