НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційні системи та технології»
спеціальності 126 «Інформаційне забезпечення робототехнічних систем»
на тему: «Система індивідуального планування участі в програмі міжуніверситетського обміну»

Виконла:
студентка IV курсу, групи ІК-81
Мірошниченко Марія Дмитрівна _____

Керівник:
ас. Кафедри ІСТ, PhD
Орленко Сергій Петрович _____

Засвідчую, що у цьому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАВДАННЯ

на індивідуальний дослідницький проєкт студентці

Мірошниченко Марії Дмитрівні

1. Тема проєкту «Система індивідуального планування участі в програмі міжуніверситетського обміну», керівник проєкту Орленко Сергій Петрович, асистент кафедри ICT, PhD.

2. Термін подання студентом проєкту: 15 червня 2022 року

3. Вихідні дані до проєкту: інформаційні джерела, технічні завдання, існуючі програмні рішення, власний досвід в предметній області.

4. Зміст пояснювальної записки:

Вступ
1. Аналіз предметної області і постановка задачі.
2. Методології та етапи розробки застосунку.
3. Вибір і обґрунтування оптимальності технічних рішень.
4. Розробка та тестування системи.
Висновки

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): діаграма прецедентів, діаграма активностей, схема бази даних застосунку

6. Дата видачі завдання 1 грудня 2021 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1	Тлумачення теми проєкту, пошук літератури	25.04.22-31.04.22	
2	Огляд існуючих рішень та постановка задачі	31.04.22-02.05.22	
3	Збір, аналіз та документування вимог	02.05.22-07.05.22	
4	Пошук та порівняння технологій для створення застосунку	07.05.22-12.05.22	
5	Планування архітектури застосунку	12.05.22-17.05.22	
6	Планування розробки ПЗ	17.05.22-21.05.22	
7	Планування тестування	21.05.22-23.05.22	
8	Оформлення дипломної роботи.	23.05.22-13.06.22	
9	Перевірка на співпадання.	15.06.2022	

Студент Марія МІРОШНИЧЕНКО

Керівник Сергій ОРЛЕНКО

АНОТАЦІЯ

Пояснювальна записка проекту складається з чотирьох розділів, містить 4 таблиці, 3 додатка та 11 джерел – загалом 66 сторінок.

Об`єкт дослідження: односторінкова веб система, що буде використовується для зберігання документів, інформування користувачів, нотування, змоги залишити повідомлення для інших.

Мета проєкту: створити систему-помічник у проходження процесу подання на програму академічної мобільності у вигляді односторінкового веб застосунку.

У першому розділі було пояснено основні терміни, проаналізовано проблеми до вирішення та існуючі рішення, задокументовано функціональні вимоги до системи.

У другому розділі було обрано методологію розробки та проаналізовано життєвий цикл проекту.

У третьому розділі було порівняно інструменти розробки: мови програмування та фреймворки, бази даних, системи управління базами даних, середовища розробки та було обрано зручні для проєкту.

У четвертому розділі було коротко описано розробку.

ABSTRACT

This project is structured in four sections, including 4 tables, 3 appendixes, and 11 references, totalling 66 pages.

Research Object: A single-store web system to be used for storing documents, providing information to users, sending notifications to others.

Objective: to create a system-assistant to the application process for the academia mobility program in the form of a one-page web-based interface.

The first section explained the basic terms, analyzed the problems to be solved and the current solutions, commented on the functional requirements for the system.

In the second section, the development methodology was reversed and the life cycle of the project was analyzed.

The third section compared development tools: programming languages and frameworks, databases, database management systems, development environment and was selected as suitable for the project.

The fourth section briefly described the development.

Пояснювальна записка

до індивідуального дослідницького проєкту

на тему: «Система індивідуального планування участі в програмі міжуніверситетського обміну»

Київ – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПЗ – Програмне забезпечення

IDE – Integrated Development Environment

SQL – Structured Query Language

JS – Java Script

GPL – General Public License

EPL – Eclipse Public License

PHP – Hypertext Preprocessor

OS – Operation System

JVM – Java Virtual Machine

GUI – Graphical user interface

MVC – model-view-controller

MTV – model-template-view

ВСТУП

Студент, що вирішив спробувати себе в навчанні в іноземному ВНЗ, стикається з багатьма ускладненнями та потребує постійних пояснень та настанов як на етапі вибору університету, так і на етапі проходження конкурсу та формування усіх необхідних документів по його завершенні.

Перед студентом будуть стояти такі задачі: визначитися з університетом для внутрішнього або міжнародного обміну, знайти його вимоги та вимоги конкурсу всередині «домашнього» університету; скласти тест з іноземної мови, якщо це необхідно і у стислі строки підготувати усі документи, що вимагає університет-партнер, відділ мобільності домашнього університету та посольство (у випадку міжнародного обміну); визначитися з дисциплінами для перезаліку за кредитами; не запізнитися з купівлею квитків та своєчасно знайти місце проживання у новому місті, якщо це необхідно.

Складно утримати в голові усі дедлайни, вимоги. Іноді проблемою стає погана проінформованість студента, і він/вона, намагаючись поєднати і навчання і оформлення документів на обмін, залишає без уваги той чи інший етап оформлення документів. У цій ситуації не вистачає помічника, що знає усі деталі кожного етапу, нагадує про найважливіше і готує до наступних кроків.

Метою даного проекту є спроектувати розробку веб-застосунку, в якому, відповівши на детальні питання щодо університету та спеціальності, на якій навчається студент, він отримував доступ до усієї необхідної інформації щодо обміну: етапи визначення, подачі документів; що слід зробити у першу чергу, що – у другу, веб-сторінки, відгуки, приклади. Маючи під рукою цей веб-застосунок, студент зможе знайти відповіді на усі питання у ньому, а не шукати у Google.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сфери дослідження

Навчання в університеті має багато можливостей, які можуть допомогти розширити свій студентський досвід. Однією з таких можливостей є програми обміну студентами. Найрозповсюдженішою такою програмою є програма Erasmus. Програма Erasmus була заснована в 1987 році як програма обміну для студентів вищих навчальних закладів. За останні 30 років програма дала 9 мільйонам людей можливість навчатися, працювати волонтером або отримати професійний досвід за кордоном.[1]

Впродовж тільки навчального року 2016/2017 більш ніж 400 000 студентів вищої освіти і персонал побували за кордоном завдяки Erasmus+[2].

Експертів дослідження Олівера Брахта з Міжнародного центру досліджень у сфері вищої освіти Університету Касселя в Германії попросили оцінити компетенції студентів учасників програм обміну після закінчення навчання в порівнянні з іншими студентами. Майже всі експерти оцінюють компетенції студентів учасників програм обміну як кращі. Знання іноземної мови та міжкультурні компетенції більшістю експертів сприймаються як набагато кращі. Найбільш вражаючим висновком є той факт, що майже всі експерти також вважають, що студенти учасники програм обміну перевершують інших студентів за соціально-комунікативними компетенціями. Багато опитаних експертів переконані, що колишні студенти Erasmus перебувають у вигіднішому становищі під час пошуку роботи, ніж випускники, які не мали міжнародної мобільності у процесі навчання. Зокрема, більшість із них вважають, що колишні студенти Erasmus мають більше шансів бути прийнятими до уваги роботодавцями як один із останніх

кандидатів. Більшість експертів також вважають, що колишні студенти Erasmus, порівняно з раніше не мобільними студентами, мають більше шансів отримати пропозицію про роботу через короткий період або за обмежених зусиль з пошуку та працевлаштуватися незабаром після закінчення навчання. [3]

Тож, як ми бачимо, програми обміну студентами – це не тільки цікава, а й важлива для подальшого розвитку у навчанні та роботі річ, якою користуються сотні тисяч студентів Європи і не тільки. Було б чудово поліпшити процес потрапляння на навчання в іншу країну за допомогою застосунку, що буде супроводжувати та наставляти напротязі цього заплутаного шляху. Такий застосунок знайде користувачів по всьому світу. Кількість учасників Erasmus+ за країною відправлення у 2014–2017 роках представлена на рисунку 1.1[2].

1.2. Пояснення термінів та досліджуваних процесів

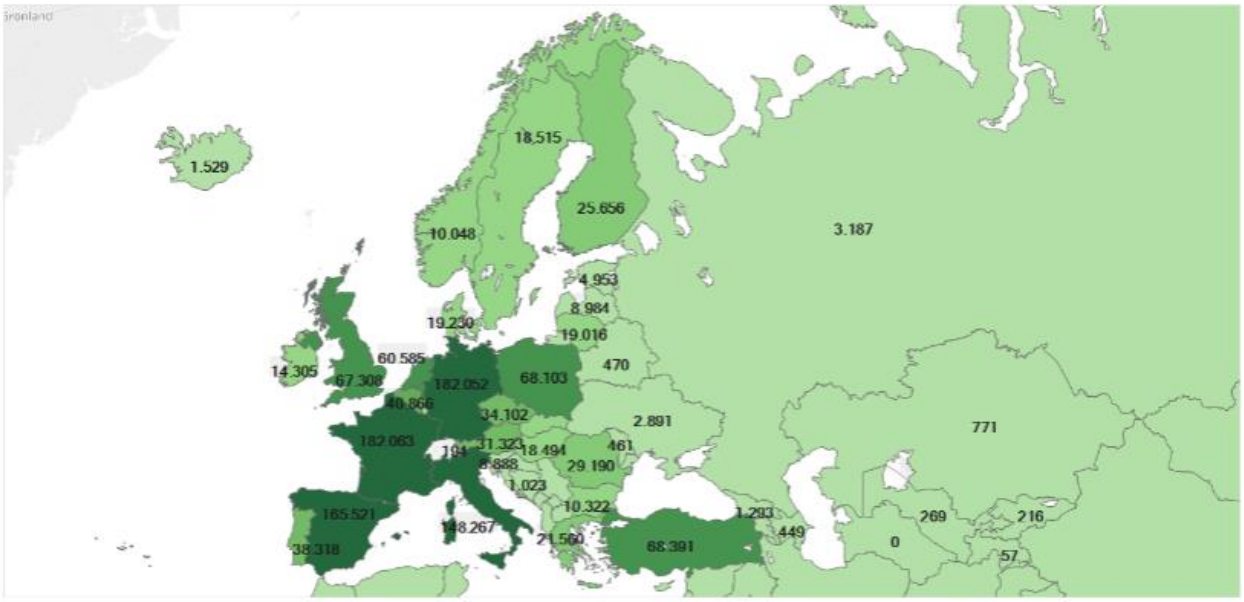
Для того, щоб розробити дійсно корисний застосунок, треба розуміти та потім у застосунку пояснити користувачу суть та деталі усіх понять, з якими стикається учасник програми обміну студентами, та усі фази, через які йому буде необхідно пройти до того, як сісти за парту в новому університеті. Необхідні терміни та поняття розглянуті нижче.

1.2.1 Поняття

Розробка додатку та його використання базуються на таких поняттях:

Академічна мобільність – це можливість студентів, науково-педагогічних працівників здійснити навчання, стажування, наукову діяльність в іншому навчальному закладі або іншому місті, країні з метою обміну науковим

досвідом. Буває внутрішньою і зовнішньою. Академічна мобільність здійснюється за допомогою програм обміну – спеціальних домовленостей між університетами. Такі програми обміну мають усі університети країни. Щоб потрапити до якогось з університетів, треба щоб



Source: Erasmus+ participant database, 2018

Рис. 1.1 – Кількість учасників Erasmus+ за країною відправлення у 2014 – 2017 роках

1) університет, в якому навчається студент, мав договір партнерства з іншим університетом. Проте є таке поняття як «індивідуальна мобільність», коли студент самостійно знаходить університет для обміну та домовляється з його представниками і повідомляє про це свій університет. Веб-застосунок, який є метою цього проекту створюється для студентів, що готуються здійснити академічну мобільність в університетах-партнерах свого «домашнього» ВНЗ.

Щоб стати учасником програми академічної мобільності, треба пройти конкурс у своєму університеті. Деякі вимоги визначає університет-партнер

(спеціальність, рівень вищої освіти, рівень володіння іноземною мовою, кількість студентів для участі в програмі), а деякі – «домашній» університет (середній бал за всі семестри навчання).

2) Академічна мобільність проводиться на базі поняття «перерахування кредитів». Учасник наперед обирає дисципліни для вивчення (якщо приймає участь як учень, не як практикант), знаходить інформацію про те, скільки кредитів «важить» кожна дисципліна для того, щоб по поверненню після програми обміну кредити були перезараховані у «домашньому» університеті.

У більшості випадків студент їде навчатися в інший університет на свою спеціальність або схожу і обирає дисципліни такі самі або схожі на ті, що вивчав би на своєму факультеті у вказаному семестрі.

3) Щоб не переривати навчання у своєму університеті на час мобільності, студент, якщо він їде саме навчатися (не на практику) до іншого університету, на своєму факультеті має можливість скласти індивідуальний навчальний план (ІНП). Це договір, підписаний завідуючим кафедрою, учасником мобільності та координатором з академічної мобільності факультету, предметом якого є узгодження питань навчання студента протягом періоду реалізації права на міжнародну академічну мобільність. В ІНП вказуються предмети, що будуть вивчатися студентом у приймаючому навчальному закладі та предмети, що будуть відповідно перезараховані на факультеті, де навчається студент після його повернення. Але якщо кількість дисциплін, обраних для вивчення у приймаючому університеті, менша за кількість предметів, що студент має здати на своєму факультеті свого університету, то всі дисципліни, що не входять до ІНП, студент має здати по поверненню або дистанційно впродовж мобільності. Якщо студент розуміє, що не зможе приборкати таку кількість навантаження, є можливість оформлення академічної відпустки.

4) Академічна відпустка – це переривання навчання зі збереженням статусу студента та здобутих успіхів до моменту, коли студент буде взмозі повернутися до навчання з того ж місця, на якому він його залишив.

5) Learning Agreement – договір, який повинен заповнювати учасник академічної мобільності, між університетом-партнером, «домашнім» університетом та студентом щодо дисциплін для вивчення, кредитів для перерахування або (у випадку навчальної чи робочої практики) – щодо кількості робочих годин.

1.2.2 Фази процесу проходження на програму обміну

Весь процес потрапляння на програму академічної мобільності за фазами:

– фаза 1 – для початку студент має переконатися, що він відповідає вимогам свого рідного університету на подачу документів на будь-яку програму мобільності;

– фаза 2 – сповнена невизначеності для тих, хто не знає наперед, у який університет якої країни хочуть потрапити. Слід обрати університет, вимоги якого виконувані та підходять студенту;

– фаза 3 – коли університет-партнер і його вимоги визначено, і учасник знає, що може себе випробувати за цими вимогами, треба подати документи на конкурс у свій університет. Під час цього процесу треба бути дуже уважним до вимог до кожного документу та дедлайнів їх подачі;

– фаза 4 – коли документи на конкурс відправлено вчасно, залишається чекати на результат і вчасно на нього відреагувати підготовкою документів для виходу наказу про направлення закордон (у випадку зовнішньої мобільності), який видає проректор університету. Під час цієї фази слід багато спілкуватися з університетом-партнером, своїм факультетом та відділом академічної мобільності свого університету;

– фаза 5 – після виходу наказу про направлення закордон (у випадку зовнішньої мобільності) слід оформлювати індивідуальний навчальний план на своєму факультеті і надіслати його міжнародний аналог Learning Agreement до університету-партнера. Або оформлювати академічну відпустку;

– фаза 6 – після або під час налагодження всіх нюансів з університетами час подавати документи на учбову візу у посольстві;

– фаза 7 – коли наказ про направлення закордон (у випадку зовнішньої мобільності) підписано, індивідуальний план сформовано і затверджено, університет-партнер чекає на студента і візу отримано, треба якнайшвидше купувати квитки і знаходити житло в країні університета-партнера;

– фаза 8 – прибувши у місце призначення, студент має зробити ряд обов'язкових дій, як то відкрити банківський рахунок, відвідати відділ мобільності свого нового університету, знайти свій факультет і записатися на відвідування навчальних предметів.

1.3 Аналіз існуючих рішень

На сьогоднішній день студент або працівник сфери освіти, який вирішив стати учасником програми міжуніверситетського обміну, аби пройти усі фази надходження на програму, звернеться, скоріш за все, до ресурсів, перерахованих і розглянутих у підпунктах нижче:

1.3.1 Офіційний сайт Erasmus

Якщо обрано програму обміну «Erasmus», учасник звернеться до її офіційного сайту, домашня сторінка якого виглядає як представлено на рисунку 1.2. Користувач тут має змогу отримати гід за програмою, як на неї

потрапити, які можливості має програма, що вона пропонує та багато іншого, але тільки з точки зору програми Erasmus. Цей сайт розповість багато корисного, але тут не знайдеш деталі про вимоги університетів та конкурсів, посольства і таке інше.

1.3.2 Сайт відділу академічної мобільності «домашнього» університету учасника

За інформацією про відкриті конкурси, вимоги до них, документи, що потребує домашній заклад освіти і таке інше учасник звернеться до сайту відділу академічної мобільності свого

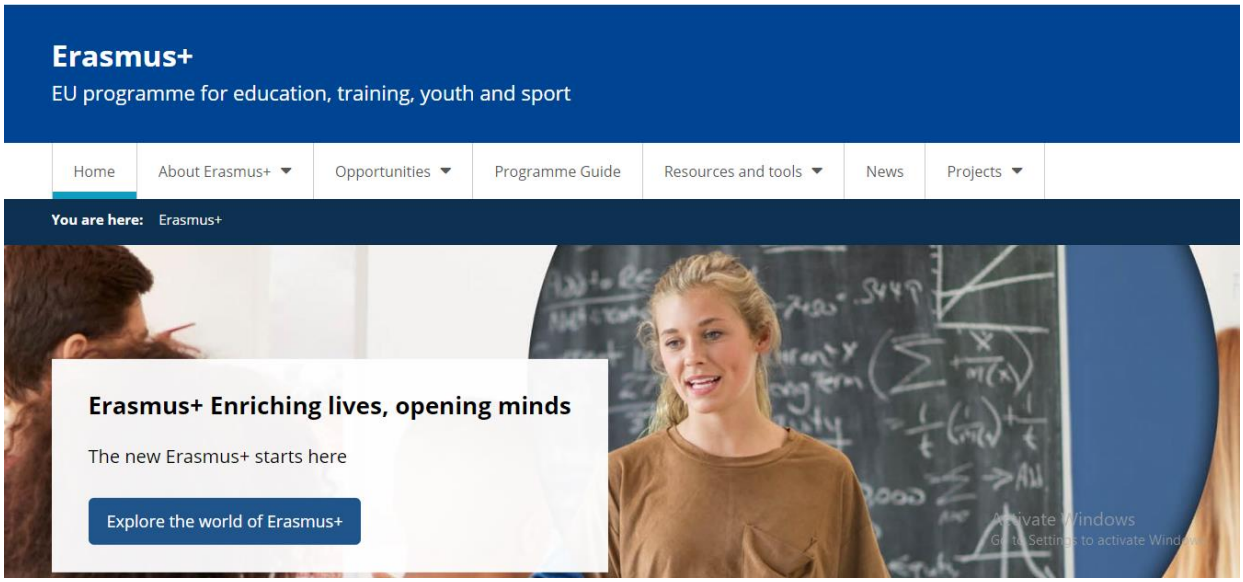


Рис. 1.2 – Домашня сторінка офіційного сайту програми Erasmus

«домашнього» університету. Розглянемо як приклад сайт відділу академічної мобільності Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Його домашня сторінка предсталена на рисунку 1.3.

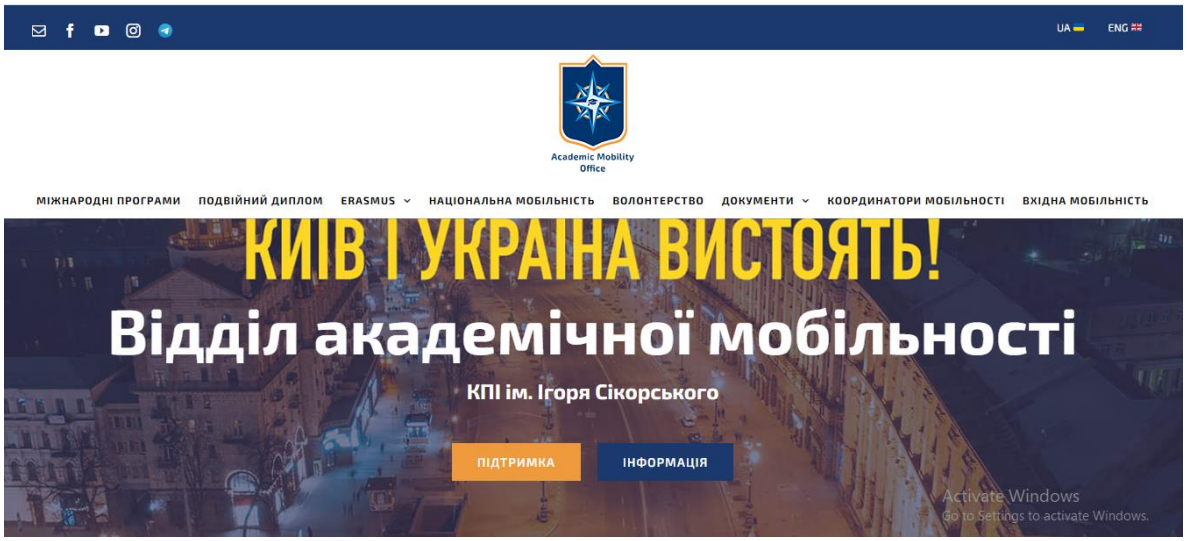


Рис. 1.3 – Домашня сторінка сайту відділу академічної мобільності НТУУ «КПІ ім. Сікорського»

Звісно, користувач знайде тут шукану інформацію про документи, які йому необхідно підготувати для конкурсу та після його проходження, про вимоги для конкурсу, координаторів мобільності факультетів і таке інше. Тут можна знайти відгуки попередніх учасників мобільності та багато корисних необхідних контактів. Але сайт відділу академічної мобільності не допоможе заповнити Learning Agreement, не нагадає про дедлайни та не підкаже що необхідно для здобуття студентської візи закордон.

1.4 Вимоги до функціоналу застосунку

У цьому пункті визначено як розроблюваний веб-застосунок може допомогти на кожній фазі, описаній вище, і які загальні функції повинні бути забезпечені.

По-перше, користувач повинен зареєструватися, мати акаунт, де буде збережено весь його досвід. У реєстраційну форму, окрім загальних питань про особу користувача, повинні бути включені питання про академічну

кар’єру, виходячи з яких система запропонує наявні варіанти для проходження конкурсу і буде допомагати на етапах заповнення документів:

– країна, регіон, місто навчання;

– університет навчання/роботи, факультет, спеціальність, курс навчання/вчений ступінь, звання, рівень вищої освіти;

– рівень володіння іноземними мовами;

– наукові публікації (їх наявність, кількість і завантаження самих публікацій у форматі, у якому їх запросять університети).

Реалізуючи першу та другу фазу, застосунок має містити посилання та добре організовану інформацію, необхідну на фазі визначення з програмою обміну та з місцем для мобільності:

– перш за все, вимоги конкурсу «домашнього» університету – курс навчання/вчений ступінь, звання, необхідний середній бал за всі роки навчання;

– університети, у які прямо зараз відкрито подачу документів на конкурс, або коли така можливість з’явиться;

– вимоги університетів-партнерів, куди зараз відкрито конкурс;

– посилання на сайти університетів, на їх учбові програми, точки на карті, де вони знаходяться;

– відгуки учасників програм мобільності;

У застосунку повинен бути передбачений переклад на будь-яку мову країн-учасників програм обміну, щоб носій будь-якої мови мав точно сформульовані, дійсні, загально визнані назви усіх документів, вулиць, учбових дисциплін, тощо.

Також застосунок має передбачити участь користувача у декількох конкурсах одночасно.

Реалізуючи третю фазу, застосунок має надати користувачу всю інформацію, яку він має знати до того, як зробить усе необхідне, щоб стати учасником конкурсу і залишиться тільки чекати його результатів, а саме:

– список документів та деделайни їх подачі з посиланням на відповідний ресурс університету;

– приклади документів, правила їх заповнення;

– якщо користувач не має сертифіката про рівень володіння іноземною мовою, то інформацію про те, чи можна пройти тест в університеті, коли і що для цього потрібно;

– пошти, на які необхідно відправляти пакет документів;

– змога зберегти дедлайни подачі та тесту у Google Calendar.

На четвертій фазі найважливіше – вчасно отримати повідомлення про результати конкурсу. Відділи академічної мобільності не завжди повідомляють учасників конкурсу щодо результатів, тож наш застосунок повинен зробити це, коли на сайті академічної мобільності факультету з'являється файл з результатами конкурсу. І якщо конкурс виграно, наступний етап – збір документів для наказу про направлення закордон (у випадку зовнішньої мобільності), що видає проректор «домашнього» університету (пункти третьої фази).

П'ята фаза – складання індивідуального навчального плану включає багато нюансів щодо вибору дисциплін, перерахування кредитів. У застосунку користувач має знайти список необхідних документів з посиланням на відповідний ресурс свого університету, а також дедлайни подачі, телефони та пошти відділів університету, нормативно-правові документи, повідомлення про наближення деделайну подачі, контакти, що можуть допомогти.

Шоста фаза – дуже серйозна, без права на помилку – взаємодія з посольством, отримання візи. Отримується, що у посольстві буде мінімум дві

зустрічі, на першу з яких треба прийти з заповненою формою на отримання візи і іншими необхідними документами та своїм фото, при зборі яких треба бути дуже уважним, і наш застосунок має надати усі нюанси для успішного подання документів, а саме:

– адреса посольства, графік роботи;

– сайт посольства;

– перелік необхідних документів;

– змога зберегти дати зустрічей у посольстві у Google Calendar.

На сьомій фазі – фазі пошуку квитків, житла – застосунок має послугувати посиланнями, інформацією щодо організацій, що займаються розміщенням закордонних студентів у різних країнах світу (таких як Interasmundo), відгуки та поради інших учасників мобільності.

На восьмій фазі, коли користувач вже прибув до місця призначення, він потребує таких підказок:

– адреси найближчих банківських відділень;

– відмітки на карті банківських відділень, офісу Еразмус, факультетів університету;

– відгуки та поради попередніх учасників мобільності щодо прибуття закордон, навчання, проживання та подорожей.

Перерахуємо також додаткові функції, які не пов’язані з окремою фазою, але можуть стати у пригоді на кожній з них:

– кожен користувач може залишити коментар на кожній фазі користування застосунком;

– застосунок має містити окрему рубрику «поради від учасників мобільності» для того, щоб учасники могли ділитися досвідом з наступними;

– застосунок має містити статті-поради, які можна буде знайти за пошуком по сторінкам застосунку;

– застосунок має надати можливість залишити зворотній зв’язок до розробника;

– має бути реалізована сторінка «записна книжка», в яку будуть вноситися справи, які користувач має зробити найближчим часом, та їх дедлайни з можливістю поставити «галочку», дописати щось своє, видалити запис, скорегувати.

За функціями застосунку, що ними може користуватися учасник програми мобільності, було розроблено діаграму прецедентів, що представлена у додатку Д1. Користувач, зареєструвавшись у системі, має змогу просто дізнатися усе необхідне про програми мобільності, що включає також і відгуки учасників, самому залишити відгук про програму, якщо він вже був її учасником, або саме мати на меті взяти участь у програмі мобільності, що включає в себе:

– дізнатись які конкурси відкрито та їх деталі;

– допомогу з оформленням документів;

– усю інформацію про зустріч у посольстві та оформлення студентської візи;

– допомогу з пошуком місця проживання в країні програми мобільності;

– інформацію про перші кроки учасника мобільності в країні призначення.

Кожен із цих прецедентів включає в себе можлиівсть занотувати якусь інформацію у записнику застосунку та потім отримати сповіщення, якщо занотована інформація містить дедлайни.

Також було розроблену діаграму активностей, представлену у додатку Д2. Діаграма показує послідовність активностей користувача, системи та бази даних у випадку, якщо користувач має на меті стати учасником програми мобільності. Діаграма описує, яка дія у застосунку відповідає кожній фазі процесу проходження на програму обміну.

1.5 Висновки до розділу

У першому розділі було розглянуто область дослідження – програми міжуніверситетського обміну, їх важливість та цікавість в житті учасників, складність у прийнятті участі в них, якими обумовлена необхідність створення застосунку-помічника. Було пояснено низку термінів та понять, з якими стикається учасник програми та потенційний користувач нашого застосунку, описано фази проходження на програму мобільності та з якими труднощами стикається майбутній її учасник. Було проаналізовано існуючі рішення – сайти, якими зараз користуються учасники конкурсу на академічну мобільність та їх незручності та недоліки. Показана виправданість створення застосунку, в якому будуть охоплені усі фази та усі деталі процесу.

Участь у програмі обміну додає професійних та комунікативних навичок її учаснику, і як наслідок, впливає на майбутнє працевлаштування. Але процедура проходження (вибір країни та університету для обміну, подання документів на конкурс, отримання його результатів, контактування та збирання документів у посольство, виїзд закордон) складна, вимагає великої уваги та має дуже багато деталей.

Веб-застосунок, функціональні вимоги до якого описано у цьому розділі, буде допомагати на кожному етапі, нагадувати про дедлайни та звертати увагу користувача на деталі. Це полегшить та спростить муторний етап підготовки до участі у програмі обміну, який виснажує учасника ще до того, як він приїде в обраний університет для навчання. У цьому розділі було вивчено кожен етап та знайдено спосіб допомоги на кожній фазі проходження на програму обміну.

2. МЕТОДОЛОГІЇ ТА ЕТАПИ РОЗРОБКИ ЗАСТОСУНКУ

2.1 Етапи розробки програмного забезпечення (ПЗ)

Кожен проект на шляху свого створення проходить через шість основних фаз, які в сумі називаються життєвим циклом проекту. Життєвий цикл проекту – це логічна схема, яка дефінізує послідовність стадій створення проекту від моменту, коли з’явилася ідея про нього до його повноцінного використання. Він називається саме циклом, тому що ніяка версія програмного продукту не може бути остаточною. Завжди є недоробки або просто з’являються нові ідеї, і цикл починається знову задля оновлень системи. Схема життєвого циклу проекту представлена на рисунку 2.1.

Рис. 2.1 – Схема життєвого циклу проєкту

Кожну фазу життєвого циклу проекту описана нижче:

– збір та аналіз вимог – від якості опису вимог залежить якість усіх наступних фаз розробки по відношенню до кінцевого результату і сам кінцевий результат. Тож має сенс підійти до цього відповідально. Після аналізу вимог слід визначитися з технічними засобами реалізації вимог. Слід спланувати усі речі, які можна слплнувати, і пам’ятати про всі речі, які спланувати не можна;

– документування вимог – має на меті визначити і прописати усі деталі;

– дизайн – на цьому етапі треба визначитися з архітектурою проекту. Обрати з поміж декількох одну найкращу;

– розробка ПЗ – уся розробка та збірка продукту відбувається тут. Програмний код, його модулі розробляються відповідно до задокументованих вимог до продукту;

– тестування – дефекти продукту відстежуються, виправляються та повторно тестуються, доки продукт не буде відповідати вимогам;

– застосування та підтримка проекту [4] – проект випускається для використання користувачами, збираються відгуки, відстежується як продукт реагує на навантаження, і з урахуванням зібраної інформації, запускається новий цикл розробки, вже націлений на доробку, оновлення основного продукту.

2.2 Методології розробки веб-додатків

При розробці сайтів, програм та інших програмних продуктів використовується досить багато відмінних методологій. Яку з них вибрати, багато в чому залежить від конкретної ситуації – особливостей проекту, бюджету, термінів, особистих уподобань розробника тощо.

Далі розглянуті п’ять основних методолгій розробки програмного забезпечення:

– "Водоспад" або каскадна модель (Waterfall Model) – це точно послідовне виконання всіх стадій розробки. Тобто нова стадія не почнеться доки не буде повністю закінчена попередня. Схема реальної роботи по розробці застосунку за каскадною методологією представлена на рисунку 2.2.

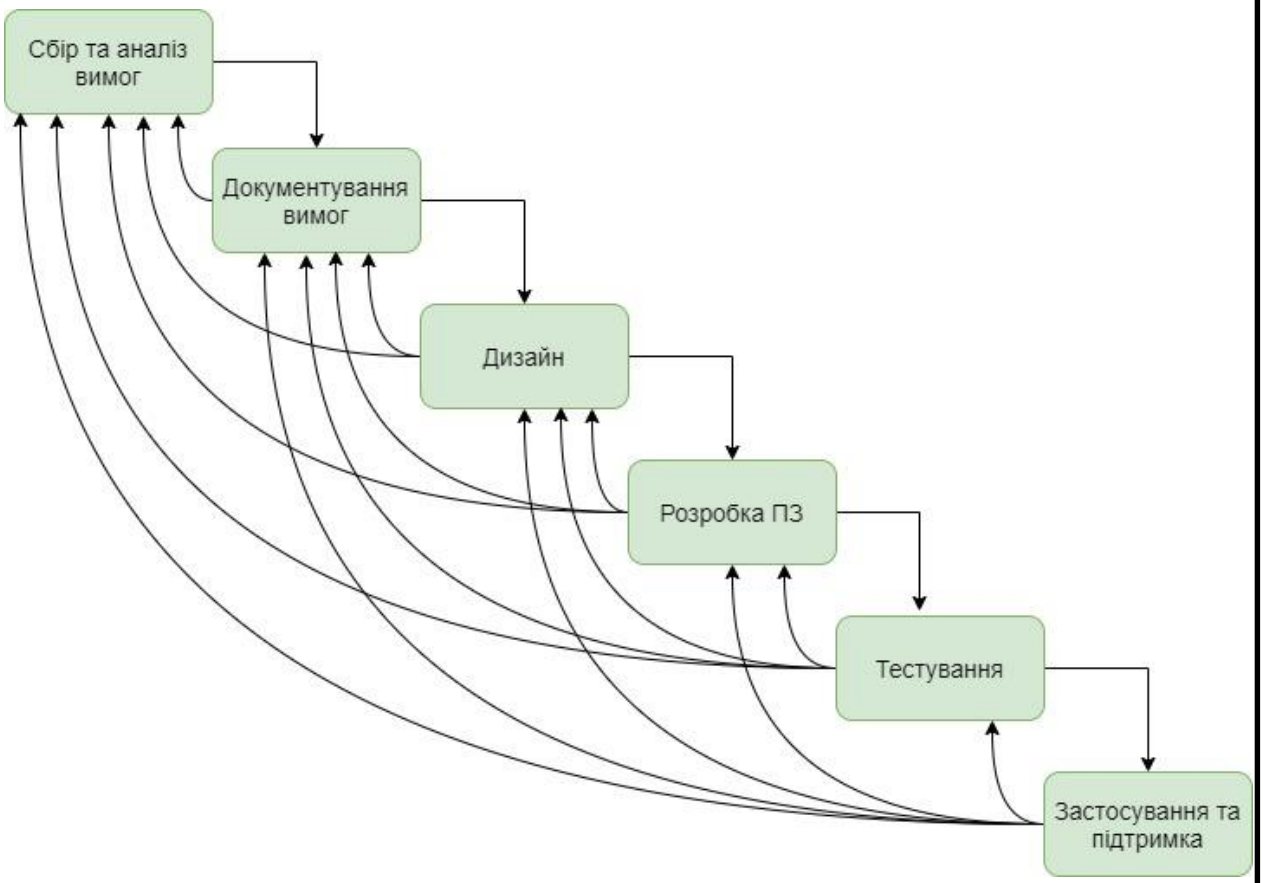


Рис. 2.2 – Схема реальної роботи по розробці застосунку за каскадною методологією

Каскадна модель дуже зручна для управління проектом, оскільки процес розробки легко відстежується. Процес розробки жорстко контролюється, що дозволяє досить точно заздалегідь визначити терміни закінчення та загальну вартість проекту. Але в реальності розробнику складно жорстко

притримуватися послідовності, тож виникає необхідність повернення на попередні фази. Каскадна модель це дозволяє, і це практикується;

– V-модель – схожа на каскадну тим, що стадії проекту йдуть у чіткій послідовності одна за одною, але відрізняється тим, що після закінчення розробки кожної стадії відбувається її тестування. Таку модель зазвичай використовують у таких системах, де не можна допустити збій у її роботі: у медицині, системах безпеки, транспортного управління тощо. Необхідна у проектах, де надзвичайно важлива стійкість системи, відсутність вразливостей;

– інкрементна модель – використовується у проектах, де розробляється декілька варіантів (збірок) готової системи. Розробка має декілька циклів, а кожен цикл – свої етапи і модулі. Для кожного модуля є свої етапи уточнення вимог, кодування, тестування. Інкремента модель називається інкрементною, тому що інкрементами називаються нові варіанти (збірки) системи, створені на базі освного варіанту (базової збірки).Цю модель використовують в проектах, де чітко прописані зміни, що повинні бути внесені в проект;

– швидка розробка додатків або «RAD Model» – це різновид інкрементної моделі, в якому компоненти проекту робить не одна команда послідовно, а декілька команд паралельно;

– гнучка модель розробки – до цієї методології належить «Scrum». Основною особливістю є те, що замовник бачить усі етапи розробки та завжди може втрутитися з поправками. [5]

Очевидно, що у випадку нашого проекту краще за все використовувати першу – каскадну – модель розробки.

2. 3 Висновки до розділу

У другому розділі було розглянуто етапи розробки застосунку. Перші два етапи: збір та аналіз вимог та документування вимог були виконані у першому розділі пояснювальної записки. У наступних розділах мова піде про третій та наступні етапи.

Було описано декілька моделей розробки програмного забезпечення визначилися з найбільш підходящою для нашого проекту. Це каскадна модель – послідовне виконання етап за етапом. Так як розробник тільки один, немає команд, які б працювали паралельно над різними етапами, каскадна модель здається найбільш реальною та зрозумілою до виконання.

3. ВИБІР І ОБҐРУНТУВАННЯ ОПТИМАЛЬНОСТІ ТЕХНІЧНИХ РІШЕНЬ

3.1 Вибір технології для реалізації проекту

Веб-застосунок – це клієнт-серверний застосунок, в якому клієнт взаємодіє з сервером через браузер. Браузер зв’язується з сервером за допомогою мережі Інтернет. Логіка роботи вед-застосунку представлена на рисунку 3.1.

Рис. 3.1 – Логіка роботи веб-застосунку

Для розробки було обрано саме веб-застосунок, тому що перевагами цього типу програмного забезпечення є:

– кросплатформеність – можливість користуватися застосунком на будь-якій операційний системі, застосунок однаково працює на Windows та Linux так як працює через інтернет браузери, що не абияк економить зусилля та час та розширює можливості користування;

– доступ з різних пристроїв – користувач може взаємодіяти з застосунком через комп’ютер, телефон планшет. Головне – доступ до мережі Інтернет;

– відсутність клієнтського програмного забезпечення – користувачу не потрібно завантажувати та потім оновлювати застосунок. Розробник змінює клієнтський інтерфейс, і оновлення до останньої версії відбудеться коли користувач перезавантажить сторінку браузера;

– безпека – веб-система має єдину точку входу, тому можна централізовано налагодити її захист. Крім того, дані користувача зберігаються в «хмарних» сховищах, тому якщо жорсткий диск зламається, інформація залишиться;

– масштабованість – навіть коли оптимізація алгоритмів да архітектури застосунку вже не допомагає системі справлятися зі зростаючим навантаженням, є можливість збільшення загальної продуктивності програми за рахунок збільшення доступних ресурсів – рознесення по декільком вузлам.

Є три основні шаблони для розробки вед-додатків:

– MPA (multi-page application) – багатосторінковий застосунок, який за кожного завантаження та перезавантаження відправляє запит на сервер та оновлює повністю всю сторінку, коли з нею роблять якусь дію;

– SPA (single-page application) – односторінковий застосунок, що має HTML-сторінку, яка динамічно, частково оновлюється в залежності від дій користувача без повного перезавантаження;

– PWA (progressive web application) – застосунок, який користувач встановлює, використовує в режимі офлайн без допомоги браузера.

Вибір відбувається між MPA (multi-page application) та SPA (single-page application), тож розглянемо більш детально переваги та недоліки кожного.

Принцип роботи SPA застосунку представлений на рисунку 3.2.

Рис. 3.2 – Принцип роботи SPA застосунку

Переваги SPA:

– швидкість перезавантаження сторінки. Так як з перезавантаженням оновлюється не вся сторінка, а тільки необхідна частина, це економить час;

– легкість створення. Для створення SPA є вже готові бібліотеки та фреймворки. Крім того, робота над frontend (зовнішнім виглядом застосунку) та backend (логікою функціонування застосунку) може вестися паралельно;

– інтерфейс. Так як веб-сторінка одна, легше наситити інтерфейс та управляти ним, створити його дружелюбним до користувача;

– кешування даних. Застосунок відправляє тільки один запит на сервер, збирає дані, а потім може функціонувати в offline-режимі;

Недоліки SPA:

– погано піддаються пошуковій оптимізації. Адреса перезавантажених сторінок практично не змінюється, а дані завантажується динамічно. Для пошукової оптимізації потрібні унікальні адреси сторінок. Пошукові боти погано вміють сканувати односторінкові веб-додатки, у результаті не всі складові застосунку зможуть з'явитися на запит у пошуковій системі;

– сильно навантажують браузер. Фреймворки frontend-частини (зовнішнього вигляду застосунку) доволі важкі;

Принцип роботи MPA застосунку представлено на рисунку 3.3.

Переваги MPA:

– проста пошукова оптимізація. Кожна сторінка має унікальну адресу, тож пошукові боти адекватно сканують їх та видають результат пошукових запитів зі сторінками MPA в топі;

– масштабованість. В MPA застосунку можна надати стільки інформації, скільки необхідно, без обмежень;

– користувачі звикли до такого типу веб-сторінок, тож навігація не викликає питань;

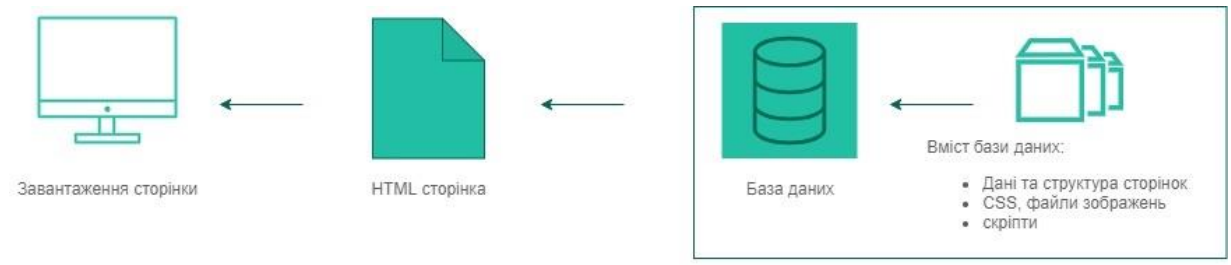


Рис. 3.3 – Принцип роботи MPA застосунку

Недоліки MPA:

– низька швидкість розробки.

Наочно різниця в принципах роботи MPA та SPA додатків представлена на рисунку 3.4.

Нижче представлена компаративна таблиця двох типів розробки вед-застосунків – таблиця 3.1.

Таблиця 3.1 – Порівняння двох технологій розробки

SPA	MPA
– простіша розробка – гнучкий інтерфейс – просте кешування даних – складна пошукова оптимізація – велика навантаженість на браузер	– низька швидкість розробки – традиційний звичний інтерфейс – масштабованість – проста пошукова оптимізація – не така висока навантаженість на браузер

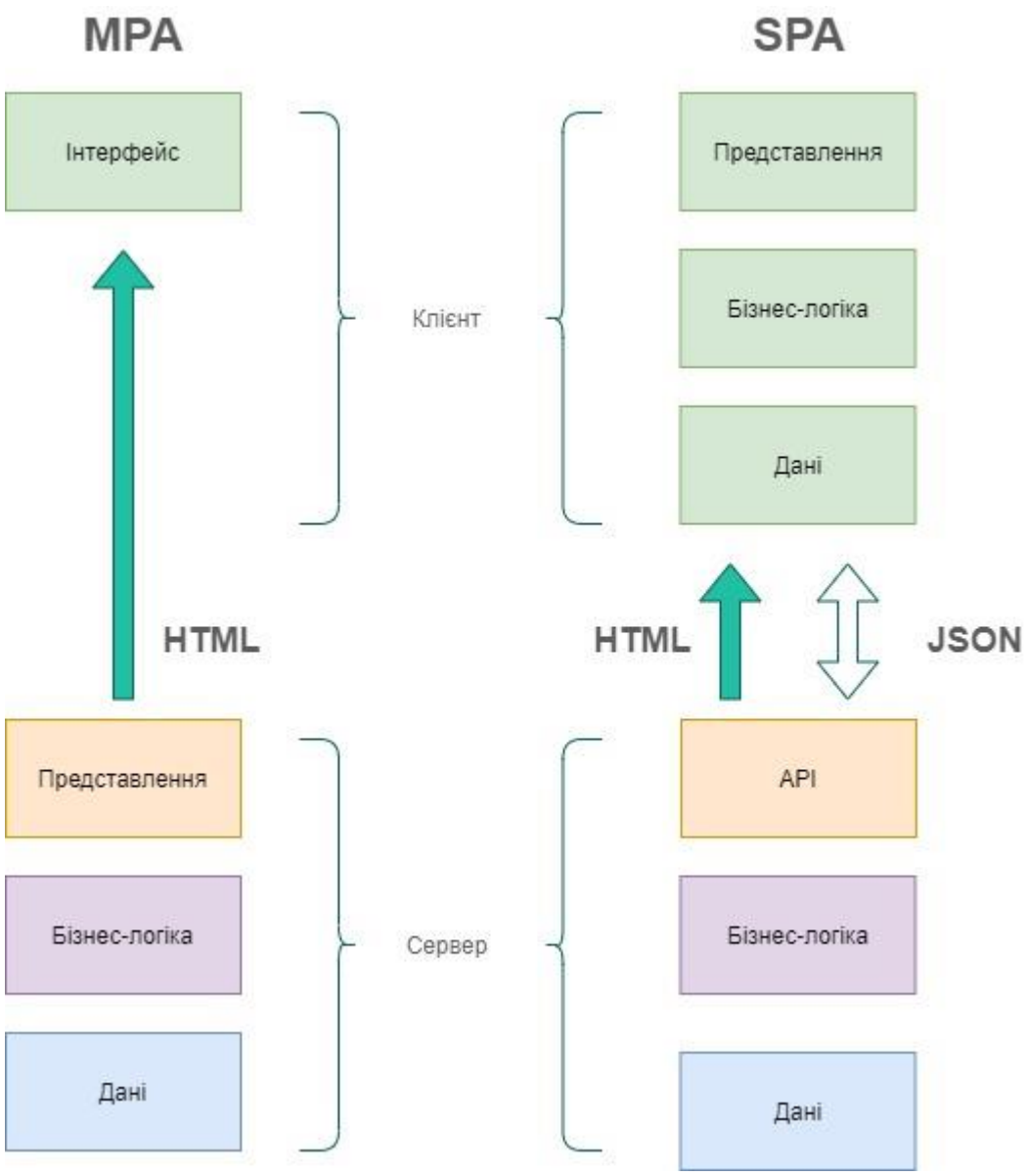


Рис 3.4 – Різниця в принципах роботи MPA та SPA додатків

Дивлячись на цілі та завдання проекту, обрано саме односторінковий веб-застосунок (SPA) до розробки. У випадку веб-застосунку для учасників програм обміну важливе саме просте кешування даних, щоб, знайшовши один раз необхідну інформацію у застосунку, знаходячись у зоні Wi-Fi, учасник зміг повернутися до неї, не маючи доступу до інтернету. Також цікава можливість створити один цільний зручний інтерфейс, а не для кожної

сторінки свій, що буде тільки плутати користувача. Проста розробка робить більш привабливою ідею створення проекту та також робить простішою його підтримку та оптимізацію.

3.2 Вибір засобів програмування

У цьому пункті обрано мови програмування, фреймворки та середовище для розробки.

Фреймворк – це програмний інструмент, який створений для поліпшення роботи над створенням додатків. Переваги використання фреймворків:

– підвищує продуктивність роботи;
– її швидкість;
– підтримує відладку;
– створює «чистий», багаторазовий, ефективний код, легкий у догляді;
– підвищує безпеку застосунку.

Типи фреймворків для веб-розробки представлено на рисунку 3.5.

Рис. 3.5 – Типи фреймворків для веб-розробки

3.2.1 Мова програмування та фреймворки

Розробка веб-застосунку поділяється на розробку backend частини та frontend частини.

Backend частина – це серверна частина веб-застосунку, програмування всього основного функціоналу, логіки застосунку, бази даних. До розробки backend-у входить програмування функціоналу та бази даних. Зараз нам доведеться обрати мови програмування та фреймворки для розробки backend частини застосунку.

Серверні або backend-фреймворки призначені для контролювання бази даних, логіки застосунку, конфігурації безпеки.

Нижче розглянуто декілька можливих та порівняємо їх.

Express – фреймворк, створений на базі Node.js – серверної платформи, базованої на JavaScript.

Переваги використання Express як фреймворку Node.js:

– дає можливість використовувати бібліотеки JavaScript у серверному оточенні;

– один і той самий код можна спільно використовувати і на клієнті і на сервері;

– JavaScript –код значно компактніше і через це виконується значно швидше;

– код клієнтської та серверної частини простіше підтримувати в узгодженому стані, так як і та і інша частини написані однією мовою;

Spring Boot – фреймворк для розробки backend-додатків, заснований на Java. Основна мета цього фреймворку – можливість формування додатків J2EE.

Переваги Spring Boot:

– Spring Boot дозволяє створювати самостійні Spring-додатки, конфігурувати їх та працювати з ними простіше;

– якщо у процесі створення застосунку виникла помилка, аналізатор помилок, що вбудований у Spring Boot, може виявити та виправити її;

– добре інтегрується з іншими фреймворками;

– Spring Boot спрощує розробку за рахунок принципу «Convention over Configuration» (цей принцип має на меті зменшення кількості рішень, що приймає розробник без втрати гнучкості та використовуючи принцип «не повторюй себе»).

Django – backend-фреймворк, написаний на Python, що маж відкритий код. Python – одна з найулюбленіших мов програмування розробників, тому Django – один із найпопулярніших серверних фреймворків.

Переваги Django:

– дозволяє без зайвих зусиль створювати динамічні додатки на Python;

– використання Django дозволяє вкрай швидко вийти на працездатний проект;

– має серйозні інструменти для впровадження та підтримки безпеки проекту: підтримує систему аутентифікації користувачів, має інструменти для захисту від різних атак;

– проекти на Django мають компактний код;

– Django – кросплатформений проект, працює на різних операційних системах;

– підтримує роботу з різними базами даних;

– проекти, створені на Django, добре піддаються масштабуванню, тож розробник може бути впевнений, що зможе розвивати свій проект коли навантаження на нього буде зростати.

Зважаючи на цілі застосунку, а також на переваги розглянутих фреймворків, обрано Django для розробки backend-частини.

Frontend частина – це розробка всього, що пов’язано з тим, що бачить користувач під час роботи з застосунком, програмування інтерфейсу, його налагодження під різні категорії екранів пристроїв.

Frontend фреймворк (фреймворк на стороні клієнта) допомагає збагатити досвід користувача при роботі з застосунком. На відміну від серверної частини, фреймворки на стороні клієнта працюють лише у браузері.

Нижче порівняно декілька найпопулярніших Frontend фреймворків. Перерахуємо їх у хронологічному порядку за часом їх створення.

Angular.js – до того як React.js набрав популярності, був найрозповсюдженішим фреймворком. Цінний своєю простотою.

Перевеги Angular.js:

– двостороння прив’язка даних. Будь-які зміни, зроблені в представлення автоматично відображаються в моделі, тобто Angular.js побудований на архітектурі MVC («Model View Controller» - модель представлення контролер). Тож розробник може швидше створювати шаблони;

– простота керування об’єктною моделлю документу. Це неможливо з іншими фреймворками JavaScript;

– несе можливість створення прототипів функцій, що допомогає швидше створити застосунок і віртуально зберігає посилання на кожну функцію;

– дозволяє створити веб-застосунок з високою швидкістю відгуку, що забезпечує швидке завантаження, плавну навігацію.

Недоліки Angular.js:

– необхідність попереднього знайомства з TypeScript;

– незручний для пошукової оптимізації так як індексування не міститься в HTML.

React.js – фреймворк JavaScript побудований на шаблоні MVC, що дозволяє ділити дані проєкту на моделі, представлення та шаблони.

Фреймворк React.js створений для того, аби розділити інтерфейс на частини і зробити розробку застосунку простіше та швидше.

Переваги React.js:

– Компонентний підхід – можливість поділити застосунок на маленькі блоки – компоненти, які залишаються придатними для редагування та масштабування після розділу.

– Оптимізація процесу оновлення.

– Має інструменти для тестування та відладки.

Недоліки React.js:

– Документація важка для розуміння новачків

– Проекти створені за допомогою React.js погано піддаються пошуковій оптимізації.

– Багато коду для простих задач.

Vue.js – відносно новий фреймфорк на базі JavaScript, призначений для покращення продуктивності та прискорення завантаження веб-сторінок та вирішення проблем, що існують у Angular та React. Одна з основних причин використання цього фреймворку великою кількістю розробників – це отримання якісного продукту за невеликі терміни та відсутність потреби у довгому навчанні.

Переваги Vue.js:

– Швидкість.

– Для Vue.js були спеціально розроблені такі бібліотеки як Vue Router та Vuex, ці інструменти не треба бути брати зі сторони та підлагоджувати під фреймворк.

– Легкий в навчанні, не потребує знань ніяких мов програмування окрім «великої трійки» – JavaScript, HTML та CSS.

– Окремі файлові компоненти з HTML, CSS та JavaScript, що робить їх легко редагуємими.

– Займає мало місця на жорсткому диску.

– Хороша документація.

– Використовує усі найкращі функції Angular та React, не маючи в собі їх недоліків.

Недоліки Vue.js:

– Відсутність масштабованості. Тобто може бути використаний тільки для невеликих додатків, які не будуть мати великого навантаження, популярності серед користувачів.

– Відсутність плагінів. Vue.js – молода технологія, яка ще не так розвинена як Angular та React, тому не має стільки плагінів, як вони. Розробнику доводиться звертатися до інших мов програмування щоб знайти вирішення проблем.

Серед перерахованих фреймворків обрано останній - Vue.js, що не потребує попередньої підготовки (окрім постійно необхідних в житті розробнибка HTML, CSS та JavaScript), з яким можна за короткий час створити закінчений проект та який має просту для розуміння доступну документацію.

3.2.2 Середовище розробки

Наступним дуже важливим вибором, який необхідно зробити є вибір середовища (IDE – Integrated development environment) для розробки.

Як критерії у виборі програмного забезпечення для розробки було використано такі поняття:

– Пропрієтарне програмне забезпечення, невільне програмне забезпечення (англ. proprietary software; від proprietary – приватне, патентоване, у складі власності та software – програмне забезпечення) – програмне забезпечення, що є приватною власністю авторів або

правовласників і не відповідає критеріям вільного ПЗ (наявність відкрито недостатньо)[6][7]. Правовласник пропрієтарного ПО зберігає у себе монополію з його використання, копіювання і модифікацію, повністю чи суттєвих моментах. Зазвичай пропрієтарним називають будь-яке невільне ПЗ, включаючи напіввільне.

– Графічний інтерфейс користувача (ГІК), (англ. graphical user interface, GUI) – інтерфейс для користувача, елементи якого виконані за допомогою графічних ображень.

– Статичний аналіз коду (англ. static code analysis) – аналіз програмного забезпечення, що виробляється (на відміну динамічного аналізу) без реального виконання досліджуваних програм. Може застосовуватися для пошуку коду, що потенційно містить вразливість[8].

– Профілювання — характеристики роботи програми, такі як час виконання окремих фрагментів (зазвичай підпрограм), число вірно передбачених умовних переходів, число кеш-промахів і т. д. Інструмент, що використовується для аналізу роботи, називають профільником або профайлером (англ. profiler). Зазвичай виконується разом із оптимізацією програми.

– Відладчик (дебаггер, англ. debugger від bug) — комп'ютерна програма чи компонент, призначений для пошуку помилок в інших програмах, ядрах операційних систем, SQL-запитах та інших видах коду. Відладчик дозволяє виконувати трасування (тобто процес покрокового виконання програми, щоб побачити послідовність виконання команд і значення змінних на даному кроці виконання програми), відстежувати, встановлювати або змінювати значення змінних у процесі виконання коду, встановлювати та видаляти контрольні точки або умови зупинки тощо[9].

Так як обрано фреймворки програмування Python Django та Vue.js, тобто обране середовище повинно підтримувати розробку на Python і JavaScript, та розробка буде відбуватися за підтримки операційної системи Windows 10.

Understand – інтегроване середовище розробки (IDE), яке підтримує статичний аналіз коду через масив візуальних, документаційних та метричних інструментів. Він був побудований, щоб допомогти розробникам програмного забезпечення зрозуміти, підтримувати та документувати їх вихідний код. Бути зрозумілим коду допомагають надання блок-схеми відносин, та побудова словника змінних та процедур з наданого вихідного коду. Переваги та недоліки цього середовища розробки представлені у таблиці 3.2.

Таблиця 3.2 – Переваги та недоліки середовища розробки Understand

Переваги	Недоліки
– Мови веб-програмування: C++ (C#, Objective C++), Java, Python, PHP, JavaScript; – платформи: Linux; Windows, Mac OS X, Solaris; – автодоповнення; – статичний аналіз коду;	– пропрієтарне ПЗ; – немає відладника (тільки інтеграція з іншими); – немає підтримки розробки GUI – немає профілювання;

Eclipse — це вільне інтегроване середовище розробки модульних програм від Eclipse Foundation. Eclipse служить насамперед платформою для розробки розширень, чим він і завоював популярність: будь-який розробник може розширити Eclipse своїми модулями. Eclipse у багатьох організаціях є корпоративним стандартом для розробки додатків через безкоштовність і високу якість. Eclipse JDT (Java Development Tools) - найвідоміший модуль,

націлений на групову розробку. Його переваги та недоліки представлені у таблиці 3.3

Таблиця 3.3 – Перевеги та недоліки Eclipse

Переваги	Недоліки
– Ліцензія: EPL (Eclipse Public License) – публічна; – Мови веб-програмування: Java, C++, Perl, Groovy, Ruby, Python, PHP; – Платформи: Windows, Mac OS X, Linux, FreeBSD, Solaris, JVM – є налагоджувач; – підтримується розробка GUI; – автодоповнення; – статичний аналіз коду; – підтримка основних фреймворків	– великовагове

NINJA-IDE (від рекурсивного акроніму: "NINJA-IDE is not another IDE") – крос-платформне інтегроване середовище розробки (IDE) для додатків мовою Python. Створено командою розробників. Переваги та недоліки цього середовища розробки представлені у таблиці 3.4.

Таблиця 3.4 – Перевеги та недоліки NINJA-IDE

Переваги	Недоліки
– Ліцензія: GPL (General	– немає профілювання;

Public License) – публічна; – Мови веб-програмування: Clojure (ClojureScript), JavaScript, Python; – Платформи: Windows, Mac OS X, Linux; – є налагоджувач; – легковагове; – підтримується розробка GUI; – автодоповнення.	– немає статичного аналізу коду.

Серед середовищ розробки, які підтримують і Python і JavaScript та працюють на Windows, вирішено обрати Eclipse.

3.2.3 База даних

Позитивний досвід користувача програми залежить від вибраного способу керування даними. Якщо програма не здатна швидко отримувати, обробляти та доставляти інформацію, то зовсім неважливо, наскільки вдалий її інтерфейс та чистий код. Більше того, всі робочі дані мають бути захищені від попадання до рук зловмисників. Щоб цього досягти, потрібно правильно підібрати систему управління базою даних.

Однією з головних проблем при виборі бази даних є вибір між структурами даних SQL (реляційна модель) і NoSQL (нереляційна модель). Обидві вони мають хорошу продуктивність, але є деякі ключові відмінності, про які слід пам'ятати.

3.2.3.1 Бази даних SQL

Реляційна база даних – це набір таблиць, між якими встановлені певні взаємозв'язки. Для обслуговування реляційної бази даних і створення запитів до неї система управління базою даних використовує мову структурованих запитів (Structured Query Language, SQL) — звичайне програмне забезпечення, що надає простий інтерфейс програмування для взаємодії з базою даних. Схему зв'язків даних реляційної бази даних можна побачити на рисунку 3.6.

Переваги:

– Реляційна база даних ідеально підходить для збереження структурованих даних (поштових кодів, номерів кредитних карток, дат, ідентифікаційних номерів). SQL – випробувана технологія: вона добре задокументована, має чудову підтримку та чудово працює з більшістю сучасних структур та бібліотек. Найбільш яскравими прикладами баз даних SQL є PostgreSQL і MySQL. Обидві зарекомендували себе як стабільні та безпечні.

– Безпека. Реляційні бази даних підтримують дозволи на доступ, які визначають, хто може читати та редагувати дані. Адміністратор бази даних може надати користувачеві права на доступ, вибір, вставку або видалення даних. Це захищає інформацію від крадіжки третіми особами.

– Використання системи управління реляційними базами даних (РСУБД) захищає дані від втрати та пошкодження завдяки чотирма властивостями ACID: атомарності, узгодженості, ізольованості та міцності

Недоліки:

– Їм не вистачає гнучкості. Реляційні бази даних не можуть ефективно працювати з напівструктурованими та неструктурованими даними, тому вони не дуже підходять для великих навантажень;

– У міру ускладнення структури даних стає все важче передавати інформацію з одного програмного рішення, орієнтованого великі дані, в інше.

– Реляційні бази даних працюють лише з одному сервері. Ці недоліки змусили розробників шукати альтернативи реляційним базам даних. В результаті з'явилися бази даних NoSQL

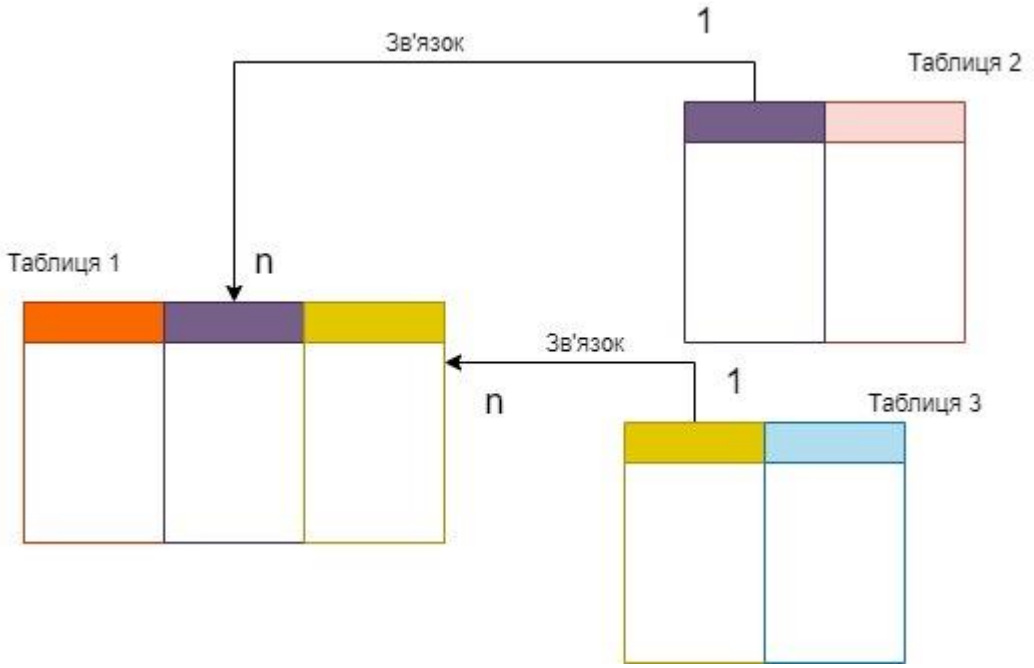


Рис. 3.6 – Схема зв'язків даних реляційної бази даних

3.2.3.2 Бази даних NoSQL

Як альтернатива реляційним баз даних було створено бази даних NoSQL (нереляційні або розподілені бази даних). Вони надають розробникам великий ступінь гнучкості та масштабованості, оскільки в них можна зберігати та обробляти неструктуровані дані (дані із соціальних мереж, фотографії, MP3-файли тощо).[10]

Дані в нереляційних базах можна змінювати динамічно, не торкаючись наявних даних. Крім того, бази даних NoSQL можуть працювати на

декількох серверах, тому масштабувати їх дешевше та простіше, ніж бази даних SQL. І оскільки бази даних NoSQL не залежать від одного-єдиного сервера, вони більш стійкі до відмови. Це означає, що у разі відмови одного з компонентів, база даних може продовжити роботу.

Бази даних NoSQL можна поділити на чотири типи (рис. 3.7).

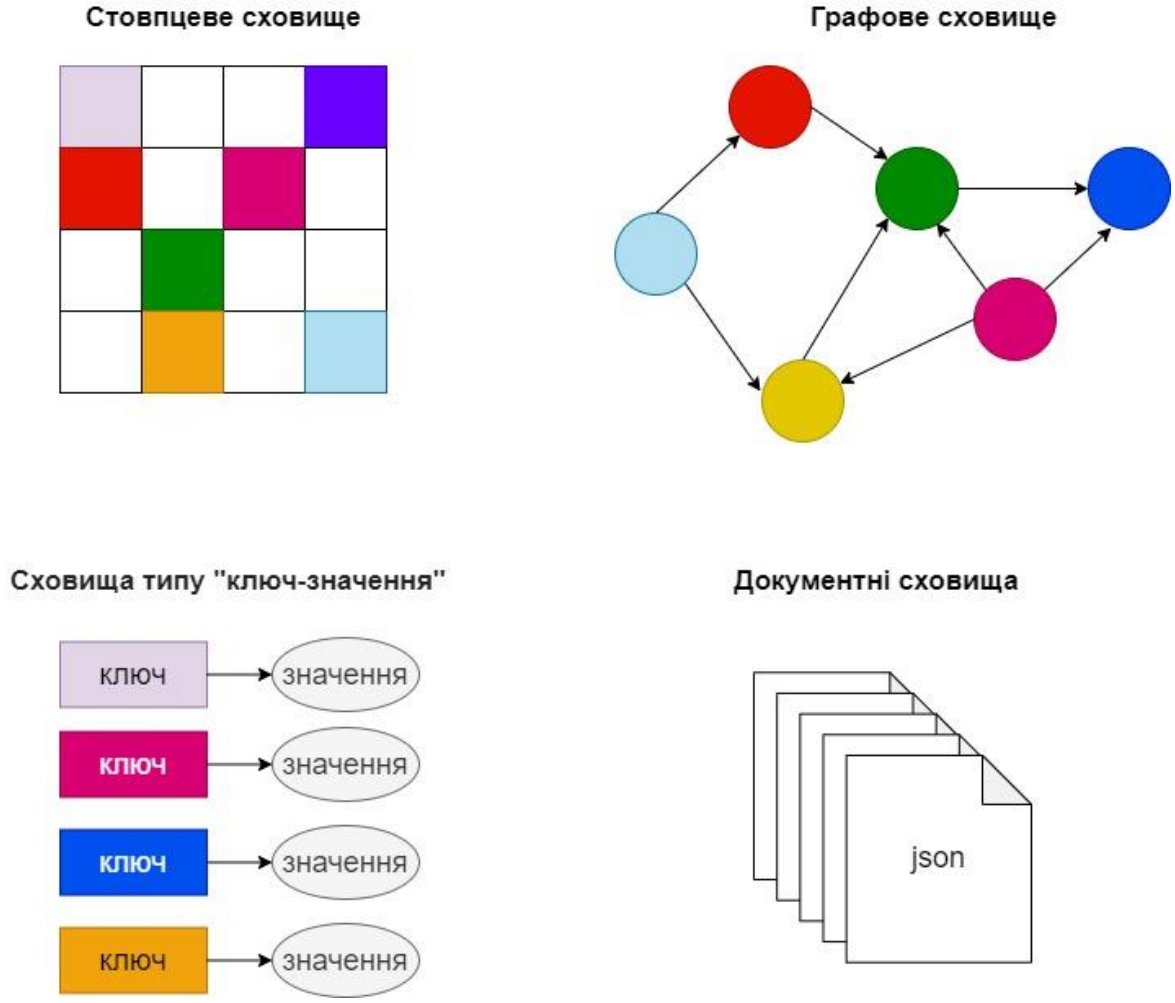


Рис. 3.7 – Чотири типи баз даних NoSQL

– сховища типу «ключ-значення» – це база даних NoSQL найпростішого типу, у якій можуть зберігатися лише пари «ключ-значення» та пропонуються базові функції для отримання значення, пов'язаного з ключем. Сховище типу "ключ-значення" - відмінний варіант, якщо ви хочете швидко

знайти інформацію за допомогою ключа. Найбільш яскравими прикладами сховищ такого типу є бази даних Amazon DynamoDB та Redis;

– документно-орієнтовані – вся інформація, що стосується певного об'єкта, зберігається в одному файлі формату BSON, JSON або XML. p align="justify"> Однотипні документи можуть бути згруповані в так звані колекції або списки. Ці бази даних дозволяють розробникам не турбуватися про типи даних та надійні зв'язки;

– стовпцеве сховище – стовпчасту базу даних оптимізовано для швидкого пошуку стовпців даних. У стовпців-орієнтованих базах даних кожен стовпець зберігається у вигляді логічного масиву значень. Бази даних такого типу забезпечують високу масштабованість та легко дублюються;

– графове сховище – графовому сховищі кожна структурна одиниця, звана вузлом, є ізольованим документом з даними довільної форми. Вузли з'єднані ребрами, що визначають зв'язок між ними. Такий підхід спрощує візуалізацію даних та аналіз графів. Графові бази даних зазвичай використовуються визначення взаємозв'язків між точками даних. Більшість графових баз даних підтримують такі функції, як пошук вузла з найбільшою кількістю зв'язків та пошук усіх зв'язаних вузлів.

Для проекту було обрано базу даних SQL.

3.2.4 Система управління базою даних

Системою управління базою даних було обрано MySQL.

MySQL – одна з найпопулярніших систем управління реляційними базами даних, яка була створена в 1995 році і зараз перебуває під керуванням Oracle. У цій системі баз даних з відкритим вихідним кодом є величезна база користувача і відмінна підтримка, і вона добре працює з більшістю структур і

бібліотек. Надається безкоштовно. Розробники можуть встановити та використовувати MySQL, не витрачаючи багато часу на налаштування.

3.3 Висновки до розділу

У третьому розділі було розглянуто, описано та порівняно технології для реалізації проекту.

Було обрано саме веб-застосунок як спосіб втілення ідеї, тому що в нашому проекти принципова крос-платформеність, відсутність необхідності встановлювати окреме програмне забезпеченя для користування застосунком.

Було порівняно multi-page application з single-page application і зроблено вибір на користь другого.

Описано backend та frontend фреймворки. Для розробки backend обрано Python Django і для frontend-розробки – Vue.js.

Середовищем розробки обрано Eclipse.

Прийнято рішення використовувати SQL-базу даних з системою управління MySql.

4. РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

4.1 Архітектура MVC та MTV

Зазвичай у додатках як наш використовується архітектура MVC. MVC (*Model*, *View*, *Controller*) - підхід до проектування застосунку, у якому код поділяють на три блоки: модель, представлення та контролер. Простими словами:

– Модель (*Model*) відподвідає за дані, які зберігаються та обробляються на сервері.

– Представлення (*View*) – це HTML-шаблон, який повертається з серверу після обробки запиту користувача, вигляд, який приймають дані на сторінці в браузері.

– Контролер (*Controller*) обробляє запити, використовує модель та контролер для реалізації необхідної реакції на дії користувача.

Схему моделі проектування MVC представлено на рисунку 4.1.

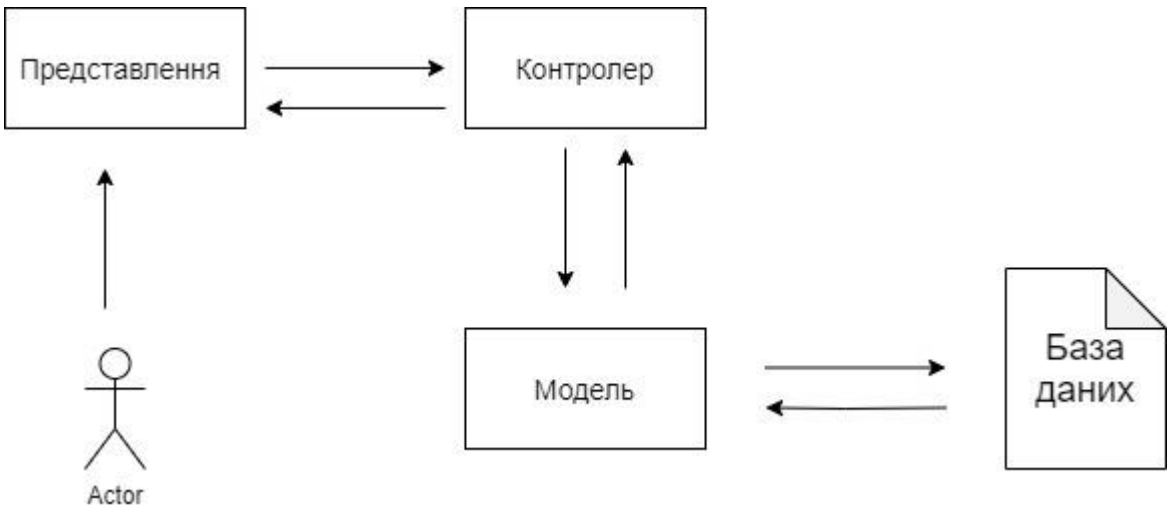


Рис. 4.1 Схема моделі проектування MVC

Але розробники Джанго вирішили трохи змінити цей спосіб проектування і створили інший – MTV (*Model, Template, View*). У їх інтерпретації:

– Представлення описує самі дані, які бачить користувач, а не те, як вони виглядають. Представлення – це функція зворотного виклику Python для визначеної URL-адреси, тому що саме функція зворотного виклику описує які дані представлені.

– Шаблон (*Template*) каже про те, як дані представлені, як їх побачить користувач у браузері.

– Контролер у випадку Django – механізм, що направляє запити до відповідного представлення у відповідності до конфігурації URL-адреси Django. [11]

4.2 Короткий опис розробки

Як вже було вказано у третьому розділі при виборі фреймворку для розробки веб-застосунку, проект Django складається з багатьох додатків. Це дозволяє поділити усі функції на блоки коду, і редагувати кожен блок знаючи, що зміни автоматично візьмуться до уваги в інших блоках.

Проект та застосунок всередині проекту створюються декількома командами у командному рядку.

Як тільки створено застосунок, пишеться представлення. За правилами Django усі представлення зберігаються у файлі views.py.

Потім щоб прив’язати представлення до URL адреси, створюється файл urls.py.

Після створення представлення підключається база даних. За замовчуванням в Django використовується SQLite. Але є способи підключити MySQL.

Відкриємо файл settings.py і подивимось зміну, що зберігає усі активні на даний момент додатки - INSTALLED_APPS:

– django.contrib.admin – застосунок адміністратора;

– django.contrib.auth – система аутентифіації;

– django.contrib.contenttypes – фреймворк для типів контенту;

– django.contrib.sessions – фреймворк для сесій;

– django.contrib.messages – фреймворк для відправки повідомлень;

– django.contrib.staticfiles – фреймворк для роботи зі статичними файлами.

Деякі з цих додатків використовують базу даних, тож треба їх встановити командою міграції python manage.py migrate. Будь-який створений застосунок слід підключати до проекту цим способом. Тоді буде встановлено увесь пакет додатків, що працюють з базою даних. Міграція змінює базу даних у відповідності до створеної моделі. Тобто спочатку модель, а потім дані з неї в базу даних.

Після цього створюється наша власна модель. Пам’ятаємо, що модель у Django займається структурою даних, з її допомогою відбувається взаємодія застосунку з базою даних. Для роботи з базами даних в проекті Django є файл settings.py, а в ньому параметр DATABASES. І також автоматично доданий при створенні застосунку файл models.py для визначення моделей. Ми змінюємо файл models.py та мігруємо зміни до бази даних.

Шаблони відповідають за зовнішній вигляд застосунку. Для їх визначення у папці проекту створюється каталог templates і в файлі settings.py вказується, що саме ця папка templates буде сховищем шаблонів. Потім у папці templates створюється файл index.html.

У файлі views.py слід вказати файл index.html та всі інші як місце для повернення результатів запитів.

У файлі urls.py у головному проекті пропишемо віповідність функції index із запитом до кореня веб-застосунку.

Але звісно в проекті буде багато додатків, багато сторінок, багато шаблонів. Для шаблонів кожного застосунку проекту створюється свій каталог.

Веб-застосунок, звісно, має статичні файли: файли стилей, скриптів, зображень. Для них у кореневу папку проекту складається каталог static. В ній для кожного типу файлів створюється окрема папка: для зображень images, для стилей – папка css тощо.

Далі за класикою визначимо стилі у файлі styles.css. І важливе - по-перше визначимо файли в шаблоні, по-друге вкажемо шлях до папки зі статичними файлами у файлі settings.py.

4.3 Опис бази даних

Розроблено схему бази даних, представлену у додатку Д3.

База даних складається з восьми таблиць та дев'яти зв'язків. Таблиці: користувачі, акаунти, зображення, програми, університети, факультети, спеціальності, документи.

Частина схеми «користувач-акаунти-зображення» відповідає за інформацію, що зберігається в акаунті про користувача та його аватар. Ці три таблиці пов'язані між собою зв'язками «один-до-одного», адже у користувача може бути тільки один акаунт і до акаунту може бути прив'язаний тільки один аватар. Таблиці «Користувач» та «Акаунти» пов’язані між собою за допомогою зовнішнього ключа AccountID, а табоиці «Акаунти» та «Зображення» - за допомогою зовнішнього ключа iconID.

Частина схеми з пов’язаними між собою таблицями «Програми-Університети-Факультети-Спеціальності» зберігає всю інформацію про університети, програми між ними, факультети та спеціальності, де ці програми доступні. Таблиці пов’язані між собою та кожна з цих чотирьох таблиць пов’язана з таблицею «Користувач» за допомогою зовнішніх ключів.

Також є таблиця «Документи», яка не пов’язана ні з якою іншою. Це таблиця прикладів документів. Вона зберігає ім’я документа, сайт, на якому можна його знайти, та сам файл документу.

4.4 Висновки до розділу

За результатами розробки повинен з’явитися застосунок, відповідний до задокументованих у першому розділі вимог. Функціональний та зручний у використанні.

ВИСНОВКИ

Була спроектована система індивідуального планування участі в программі міжуніверситетського обміну.

В першому розділі були проаналізовані сфери дослідження та проблематики питання, доведенню важливості проєкту та попиту на застосунок. Були визначені деякі терміни та поняття, необхідні для розуміння, визначені та описані фази та їх проблеми, які проходить та з якими стикається учасник програми академічної мобільності. Були знайдені та проаналізовані електронні інформаційні ресурси, якими буде користуватися учасник програми академічної мобільності, який не має помічника у вигляді нашого застосунку, і був зроблений висновок, що інформацію з них необхідно та корисно централізувати та хронологізувати за порядком використання учасником мобільності. Було також визначено та задокументовано вимоги до функціоналу системи.

У другому розділі перейдено до питань розробки більш детально, розглянуто з чого складається життєвий цикл проєкту і так визначив етапи розробки. Також було розглянуто декілька можливих методологій розробки програмного забезпечення і знайдена оптимальна модель.

Задача третього розділу у тому, щоб знайти найкращі технологічні рішення для розробки проєкту: технологію, засоби програмування (мову програмування, фреймворки), середовище розробки, вид бази даних та систему управління нею. Були порівняні технології та знайдено оптимальне рішення.

Четвертий розділ пояснює архітектуру застосунку та коротко розглядає розробку застосунку.

ПЕРЕЛІК ПОСИЛАНЬ

1. «Insights into the expectations of mobility students: the impact of Erasmus in their future professional careers» (2017) [Електронний ресурс]. Режим доступу:https://d1wqtxts1xzle7.cloudfront.net/85999881/3064-libre.pdf?1652689851=&response-content-disposition=attachment%3B+filename%3DInsights_into_the_expectations_of_mobili.pdf&Expires=1652864950&Signature=YBAkjJhTMVhAJ2XxTvRyiHcR-UhwlIsrmkVhy6oplgr2leH-xv58F2N4iQAQ6jNtjg9ZYH71Tn0WIg5Ay26M7k2r6LYMlIjGCsNwPve7L39aAgWB1dLOq985qBpdLsbcbEC

2. «Erasmus+ Higher Education Impact Study» Manuel Souto-Otero (2019) [Електронний ресурс]. Режим доступу: https://op.europa.eu/en/publication-detail/-/publication/94d97f5c-7ae2-11e9-9f05-01aa75ed71a1/language-en

3. «The Professional Value of ERASMUS Mobility» Oliver Bracht (2006) [Електронний ресурс]. Режим доступу: https://www.eurashe.eu/wp-content/uploads/2022/02/WG4-R-Professional-value-of-ERASMUS-mobility-Teichler.pdf

4. «Что такое жизненный цикл разработки» (2021) [Електронний ресурс]. Режим доступу : https://vc.ru/u/700268-marketing-solveit/199225-chto-takoe-zhiznennyy-cikl-razrabotki-po-i-kakie-problemy-voznikayut-na-kazhdom-etape-sdlc

5.«Методы разработки веб приложений и сайтов» Д. А. Дроздович [Електронний ресурс]. Режим доступу: https://libeldoc.bsuir.by/bitstream/123456789/44365/1/Drozdovich_Metody.pdf

6. «Правова природа Угоди з торговельних аспектів прав інтелектуальної власності» Чибісов Д.М. (2012). [Електронний ресурс].

Режим доступу: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/14415/1/%D0%9F%D0%A0%D0%90%D0%92%D0%9E%D0%92%D0%90%20%D0%9F%D0%A0%D0%98%D0%A0%D0%9E%D0%94%D0%90%20%D0%A3%D0%93%D0%9E%D0%94%D0%98%20%D0%9F%D0%A0%D0%9E%20%D0%A2%D0%9E%D0%A0%D0%93%D0%9E%D0%92%D0%95%D0%9B%D0%AC%D0%9D%D0%86%20%D0%90%D0%A1%D0%9F%D0%95%D0%9A%D0%A2%D0%98%20%D0%9F%D0%A0%D0%90%D0%92.pdf

7. «Freedom for Users, Not for Software» Benjamin Mako Hill (2013) [Електронний ресурс] Режим доступу: https://mako.cc/writing/hill-freedom_for_users.html

8. Benjamin Livshits «Improving Software Security with Precise Static and Runtime Analysis», section 7.3 "Static Techniques for Security," Stanford doctoral thesis, 2006. [Електронний ресурс]. Режим доступу: http://research.microsoft.com/en-us/um/people/livshits/papers/pdf/thesis.pdf

9. "Debuggers for Programming Languages" Sanjeev Kumar Aggarwal and M. Sarath Kumar (2003). [Електронний ресурс]. Режим доступу:https://www.researchgate.net/publication/255665336_Debuggers_for_Programming_Languages

10. «Ключевые аспекты при выборе базы данных для вашего приложения» (2021) [Електронний ресурс]. Режим доступу:https://habr.com/ru/company/otus/blog/562852/

11. «Django appears to be a MVC framework, but you call the Controller the “view”, and the View the “template”. How come you don’t use the standard names?» [Електронний ресурс]. Режим доступу: https://docs.djangoproject.com/en/dev/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names

ДОДАТКИ

Номер рядка	Формат	Позначення	Найменування	Кількість листів	Примітка
1			Документація загальна		
2					
3			Знову розроблена		
4					
5	А4	ІК81.150БАК003 ПЗ	Пояснювальна записка	57	
6	А3	ІК81.150БАК003 Д1	Система індивідуального	1	
7			планування участі в програмі		
8			міжуніверситетського обміну		
9			Діаграма прецедентів.		
10	А3	ІК81.150БАК003 Д2	Система індивідуального	1	
11			планування участі в програмі		
12			міжуніверситетського обміну		
13			Діаграма активностей		
14	А3	ІК81.150БАК003 Д3	Система індивідуального	1	
15			планування участі в програмі		
16			планування участі в програмі		
17			Схема бази даних		
18					
19					
20					

Зм.	Лист	№ докум.	Під	Дата	**ІК81.150БАК003 ТП**		
Розробив		Мірошниченко			Система індивідуального планування участі в програмі міжуніверситетського обміну Відомість проєкту	Літ.	Арк. 1 / Аркушів 1
Перевірив		Орленко С.					
						КПІ ім. Ігоря Сікорського ФІОТ Група ІК-81	
Затв.		Ролік О. І.					