Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200137.01.09.ПЗ

Виконав студент IV курсу, група ІПЗ-19-1 _____ І. І. Зозуля
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Г. І. Радельчук
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент _____ Г. І. Радельчук
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

19. 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
05 02 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Зозулі Іллі Івановичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків

Керівник роботи Радельчук Галина Іванівна, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.

3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування довідково-інформаційної системи, програмна реалізація та тестування системи

5. Перелік креслень (із зазначенням обов'язкових креслень) Три креслення (UML-діаграма варіантів використання, UML-діаграма послідовності, UML-діаграма діяльності)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Радельчук Г. І., доцент кафедри ІПЗ	10.06.23	16.06.23
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ		

7. Дата видачі завдання « 02 » січня 2023р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КвР), визначення та узгодження індивідуальних тем КвР	01.12 – 30.12.2022	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення. Розробка графічної частини.	01.02 – 28.02.2023	
4 Програмна реалізація	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2023	
7 Попередній захист КвР	Травень 2023 (згідно графіка)	
8 Перевірка КвР на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2023	
9 Підготовка до захисту та захист КвР	з 01.06.2023	

Студент _____ І. І. Зозуля
Підпис Ініціали, прізвище

Керівник роботи _____ Г. І. Радельчук
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків».

Автор роботи: Зозуля Ілля Іванович.

Керівник роботи: Радельчук Галина Іванівна.

Пояснювальна записка: 66 с., 19 рис.,3 табл., 3 дод., 43 джерела.

Графічна частина: 3 креслення у ф. А3.

ДОВІДКОВО-ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, КІНОТЕАТР, JAVASCRIPT, VUE.JS, HTML, CSS, SCSS, PHP.

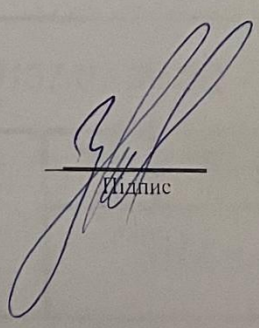
Метою роботи є розробка довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків.

У кваліфікаційній роботі проведена робота над розробкою довідково-інформаційної системи для кінотеатру з реалізацією онлайн-продажу квитків.

У рамках дослідження предметної області були проаналізовані існуючі додатки та їхні функціональні можливості, а також виявлені недоліки та проблеми, що можуть бути враховані при розробці нової системи. Для реалізації проекту було обрано фреймворк Vue.js, оскільки він забезпечує швидку та ефективну розробку користувацького інтерфейсу, а також надає зручні інструменти для керування станом додатку та навігацією між сторінками.

Система базується на фреймворку Vue.js з використанням Vue Router та Vuex для керування станом застосунку. Система надає можливість користувачам переглядати розклад сеансів, отримувати інформацію про фільми, реєструватися та придбавати квитки онлайн. В роботі також проаналізовано та обрано оптимальні інструменти для розробки та реалізації функціоналу. Результатом роботи є готовий програмний продукт, який сприятиме зручності та ефективності процесу купівлі квитків у кінотеатрі.

16.06.23
Дата


Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200137.01.09.ПЗ	Пояснювальна записка	66		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічна частина			
4	А3	КвРІПЗ.200137.01.09.Е8	UML-діаграма класів	1		
5	А3	КвРІПЗ.200137.01.09.Е8	UML-діаграма	1		
			послідовності			
6	А3	КвРІПЗ.200137.01.09.Е8	UML-діаграма	1		
			діяльності			

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200137.01.09.ВД		
Виконав		Зозуля І.І.		16.06	Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків	Літ.	Арк. 1 / Акрушів 1
Керівник		Радельчук Г.І.		16.06			
Н. Контр.		Радельчук Г.І.		16.06	Відомість документів	ХНУ, ІПЗ-19-1	
Зав. Каф.		Бедратюк Л.Д.		19.06			

ЗМІСТ

ВСТУП

Кінотеатри є одним з найбільш популярних місць розваг, де люди можуть насолоджуватися переглядом фільмів у затишній атмосфері разом з друзями та родиною. Проте, з огляду на сучасні технології та швидкий темп життя, багато людей віддають перевагу онлайн-продажу квитків, який дозволяє їм заощадити час та зручно обрати місце у кінотеатрі. З метою поліпшення сервісу та розширення аудиторії, досить важливою є розробка програмного забезпечення для кінотеатрів, яке б дозволяло забезпечувати ефективний та зручний онлайн-продаж квитків, а також надавати користувачам інформацію, розклад сеансів та інші деталі, пов'язані з перебуванням у кінотеатрі.

У даній кваліфікаційній роботі буде розглянуто розробку довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків на основі Vue js з використанням api Vue Router та Vuex. Головною метою розробки є покращення сервісу для користувачів кінотеатру та забезпечення зручного та ефективного онлайн-продажу квитків. Дане програмне забезпечення забезпечить не тільки продаж квитків, але й можливість перегляду інформації про фільми.

Дана кваліфікаційна робота буде складатися з кількох етапів, а саме: дослідження предметної області та постановка задачі, проектування системи, розробка програмного забезпечення, тестування та валідація розробленої системи, а також аналіз результатів та формулювання висновків. Основна увага буде приділена детальному опису функціональності системи, в тому числі онлайн-продажу квитків, відображенню інформації про фільми та сеанси, можливості пошуку та фільтрації інформації. Буде проведено огляд та аналіз подібних існуючих рішень на ринку, а також визначені технічні та функціональні вимоги до системи.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний наліз предметної області, її структурних та функціональних особливостей

Змістовний аналіз предметної області є важливим етапом дипломної роботи, оскільки він дозволяє детально розглянути та проаналізувати основні аспекти, структуру та функціональні особливості досліджуваної області.

Ця тема включає в себе детальне дослідження структури та функціональних особливостей цієї області. Перше - аналізується структура кінотеатру, включаючи залі, сеанси, розклад фільмів, ціни на квитки, розташування та кількість місць у залах, технічне обладнання та інші складові. Це дозволяє зрозуміти, які фактори впливають на функціонування кінотеатру та як вони пов'язані між собою. Друге - розглядається процес онлайн-продажу квитків, включаючи систему бронювання та оплати, взаємодію з платіжними шлюзами, електронні квитки та їх доставку клієнтам. Досліджується, які функції та процеси пов'язані з онлайн-продажем квитків та як вони взаємодіють з іншими складовими системи кінотеатру. Третє - розглядається структура та функціонал інформаційної системи, яка використовується для керування кінотеатром та онлайн-продажу квитків. Вивчається база даних, система керування контентом, модулі адміністрування та звітності, інструменти аналітики та інші складові системи. Аналізується взаємозв'язок між різними компонентами інформаційної системи та їх вплив на функціональність кінотеатру. Та останнє - досліджуються потреби та вимоги різних категорій користувачів системи, включаючи власників кінотеатру, адміністраторів, клієнтів (відвідувачів кінотеатру) та операторів підтримки. Аналізуються їх взаємодія з системою та функціональність, яка задовольняє їх потреби.

Змістовний аналіз дозволяє зрозуміти основні характеристики та особливості предметної області "Довідково-інформаційна система кінотеатру з

реалізацією онлайн-продажу квитків". Він надає базові знання для подальшого проектування та розробки системи, враховуючи потреби користувачів та особливості функціонування кінотеатру. Додатково, змістовний аналіз дозволяє виявити прогалини або проблемні місця в існуючих рішеннях, що може послужити основою для подальших досліджень та вдосконалення системи.

Кінотеатр є основною складовою цієї предметної області. Аналізуючи його структуру, ми розглядаємо різні аспекти, такі як залі, сеанси, ціни на квитки та їх доступність, а також технічне обладнання. Досліджуючи ці аспекти, ми отримуємо глибоке розуміння того, як кінотеатр функціонує та як впливають на нього різні фактори.

Онлайн-продаж квитків є ще однією важливою функціональною особливістю даної предметної області. Аналізуючи процес продажу квитків в Інтернеті, ми досліджуємо систему бронювання та оплати, електронні квитки та їх доставку. Це включає в себе аналіз взаємодії з платіжними шлюзами та забезпеченням безпеки транзакцій.

Окрім цього, ми аналізуємо інформаційну систему, яка використовується для керування кінотеатром та онлайн-продажу квитків. Це включає розгляд бази даних, модулів адміністрування та звітності, інструментів аналітики та інші компоненти системи. Аналізуючи цю систему, ми розуміємо, як вона організована та які функції вона забезпечує для кінотеатру та його клієнтів.

У результаті змістовного аналізу предметної області ми отримуємо повний огляд кінотеатру як бізнесу, його взаємодії з клієнтами та особливостей онлайн-продажу квитків. Це дозволяє нам розуміти потреби користувачів та визначати функціональні вимоги до розробки довідково-інформаційної системи, що сприятиме оптимізації процесів управління та забезпечить зручність та задоволення для клієнтів.

1.2 Аналіз наявного програмного забезпечення предметної області

На сьогоднішній день існує багато додатків, що надають можливість онлайн-бронювання та покупки квитків на кіносеанси. Один з найбільш відомих сервісів цього типу - це Fandango, що пропонує широкий вибір кінотеатрів та фільмів в США та Канаді. Сервіс надає можливість здійснювати онлайн-покупки квитків, а також переглядати трейлери та огляди фільмів.

Ще одним відомим сервісом є IMDb (Internet Movie Database), (рисунок 1.1), який є одним з найбільш відвідуваних сайтів про кіно та телебачення.

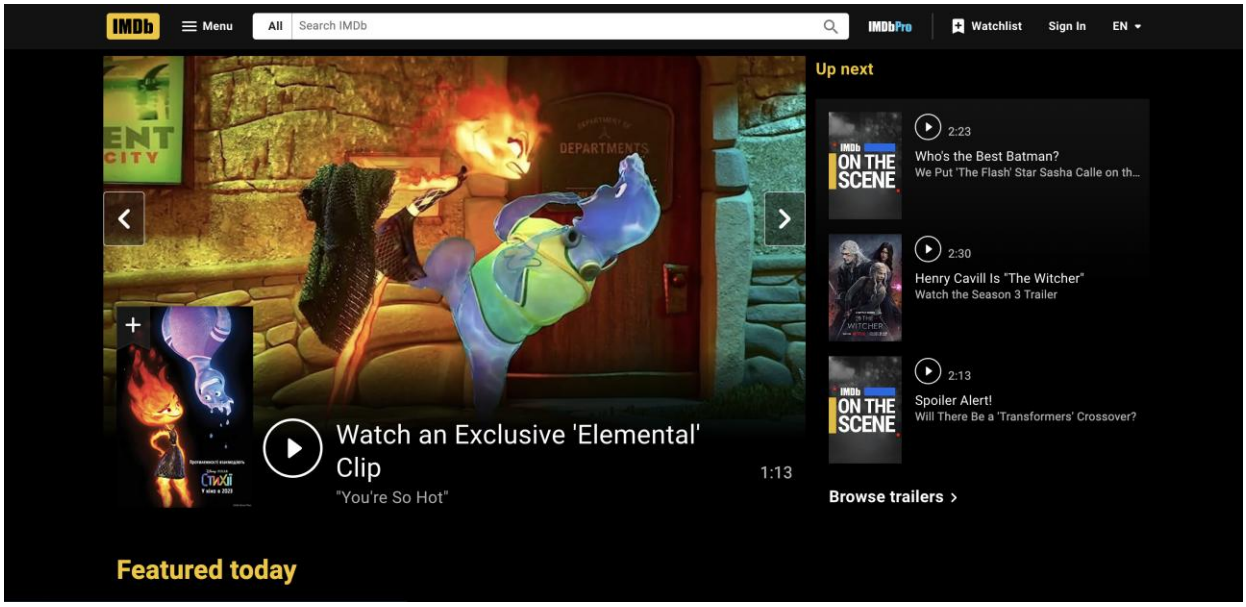


Рисунок 1.1 – Internet Movie Database – головна сторінка [1]

IMDb надає користувачам можливість переглядати інформацію про фільми, серіали та акторів, а також огляди та рейтинги. Крім того, сервіс має функціонал онлайн-бронювання квитків на кіносеанси в деяких кінотеатрах.

Іншим відомим сервісом є Rotten Tomatoes (рисунок 1.2), який надає огляди та рейтинги фільмів, а також можливість здійснювати онлайн-бронювання квитків на кіносеанси. Сервіс також має функціонал пошуку кінотеатрів, які мають сеанси для обраних фільмів.

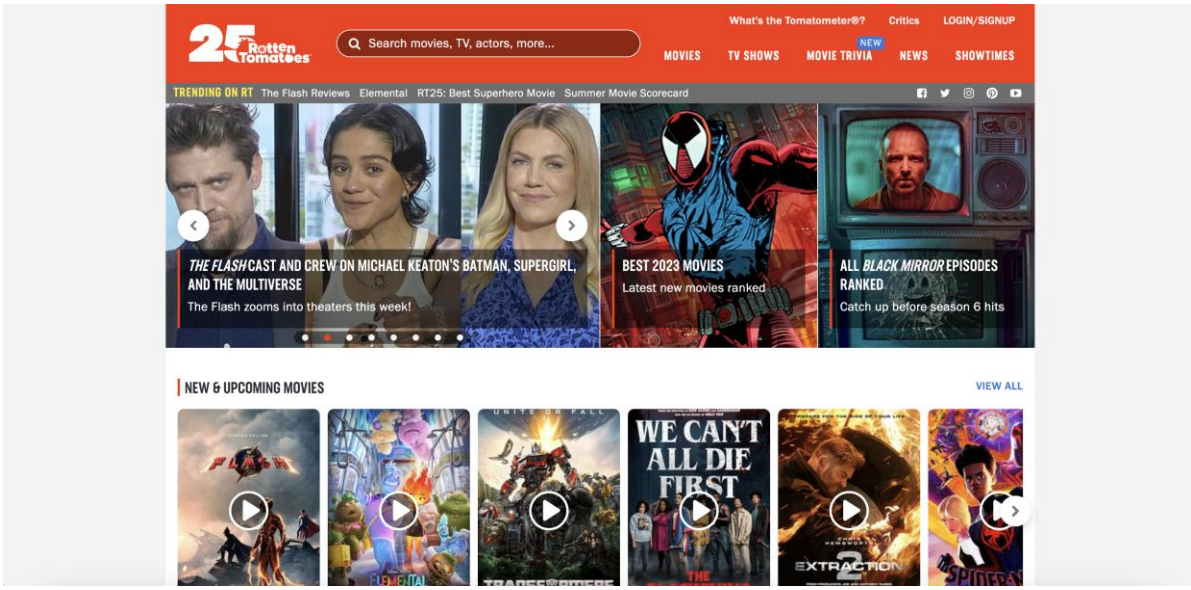


Рисунок 1.2 – Rotten Tomatoes – головна сторінка [2]

З урахуванням особливостей та потреб користувачів, було вирішено розробити довідково-інформаційну систему кінотеатру з реалізацією онлайн-продажу квитків на платформі Vue.js. В порівнянні з існуючими додатками, розроблена система буде мати переваги відносно локальних кінотеатрів, що не мають можливості надавати інформацію про фільми та квитки онлайн.

Також, додаток буде мати простий та зручний інтерфейс для користувачів, з можливістю швидкого та зручного пошуку фільмів та сеансів в кінотеатрі.

Узагальнюючи, дослідження предметної області дало змогу зрозуміти потреби та вимоги користувачів у галузі онлайн-продажу квитків на кіносеанси, а також виявити переваги та недоліки існуючих додатків. Це дозволило планувати та розробляти систему, що відповідає найкращим стандартам та вимогам користувачів.

У процесі дослідження було виявлено кілька популярних додатків, які використовуються в кінотеатрах. Деякі з них надають інформацію про розклад сеансів, опис фільмів та можливість купувати квитки онлайн. Інші додатки можуть мати додатковий функціонал, такий як відгуки та рейтинги фільмів, рекомендації, програми лояльності та акції.

Проаналізувавши ці додатки, було виявлено кілька недоліків. Деякі з них мають складний та заплутаний інтерфейс, що може ускладнювати користування для непрофесійних користувачів. Деякі системи мають обмежену функціональність або недостатньо розширюваність для впровадження нових можливостей. Також виявлено проблеми зі швидкістю завантаження сторінок та відсутністю гнучкості при налаштуванні фільтрів та пошуку.

Враховуючи ці недоліки, було вирішено розробити нову довідково-інформаційну систему кінотеатру, яка має зручний та інтуїтивно зрозумілий інтерфейс, багатий функціонал та швидку реакцію на дії користувачів.

Деякі системи працюють в парі з платіжними шлюзами, що дозволяють користувачам зручно та безпечно придбати квитки через Інтернет. Проте, деякі додатки мають обмежені можливості у плані підключення до різних платіжних систем або надання різних варіантів оплати. Отже, вирішено розробити систему, що забезпечить широкі можливості для онлайн-продажу квитків, включаючи підтримку різних платіжних систем та різні варіанти оплати.

Крім того, під час аналізу було виявлено, що деякі додатки не надають достатньої інформації про фільми, таку як опис, трейлери, рейтинги тощо. Це може ускладнювати вибір фільму для користувача і обмежувати можливості кінотеатру привернути більше глядачів. Тому важливим аспектом розробки нової системи було забезпечення повної та актуальної інформації про фільми, що включає опис, трейлери, рейтинги, відгуки та інші деталі, які допоможуть користувачам прийняти рішення щодо перегляду фільму.

Додатковою проблемою, яку було виявлено під час аналізу існуючих додатків, є відсутність гнучкості та налаштовуваності фільтрів та пошуку. Деякі системи обмежують користувачів у виборі фільтрів або не надають можливості збереження налаштувань пошуку.

1.2 Визначення вимог до довідково-інформаційної системи

Визначення вимог є важливою частиною процесу розробки довідково-інформаційної системи для кінотеатру. Функціональні вимоги: Це конкретні функції та можливості, які система повинна мати. Наприклад, можливість перегляду розкладу сеансів, пошуку фільмів, реєстрації користувачів, онлайн-продажу квитків. Важливо чітко визначити всі функціональні вимоги системи.

Основні вимоги проекту включають:

1. Розробка інтерфейсу для користувачів: Реалізація зручного та інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачам легко переглядати інформацію про фільми, доступні сеанси, місця та ціни на квитки. Користувачі повинні мати можливість швидко та зручно забронювати та придбати квитки в обраний кінотеатр

2. Реалізація онлайн-продажу квитків: Розробка механізму онлайн-продажу квитків з можливістю вибору місць та розкладу сеансів. Система повинна забезпечувати безпеку та захист персональних даних користувачів під час транзакцій.

3. Управління контентом: Розробка адміністративної панелі, що дозволяє керувати контентом системи, включаючи додавання, редагування та видалення фільмів, сеансів, цін на квитки та іншої інформації. Адміністратори повинні мати можливість легко оновлювати та модифікувати дані системи.

4. Забезпечення безпеки та захисту: Розробка системи з урахуванням захисту персональних даних та конфіденційності користувачів. Система повинна забезпечувати безпеку платіжних транзакцій та інших процесів обробки даних.

Отже, основна мета проекту полягає у створенні функціональної та ефективної довідково-інформаційної системи для кінотеатру, яка забезпечує зручний та безпечний онлайн-продаж квитків, управління контентом та підтримку різних мов та культур.

Розроблена система дозволить користувачам швидко та зручно дізнатися інформацію про розклад сеансів, купити квитки в будь-який зручний для них час та місце, а також забезпечить можливість вибору місць у залі за допомогою інтерактивної схеми. Крім того, система дозволить адміністраторам кінотеатру зручно та ефективно керувати контентом на сайті та моніторити продажі квитків. Розроблена система також забезпечить захист персональних даних користувачів та безпеку платіжних транзакцій.

Важливою метою проекту є також підтримка різних мов та культур, що дозволить користувачам з різних країн та регіонів використовувати систему на їхніх власних мовах. Крім того, розроблена система має забезпечувати максимальну продуктивність та швидкість роботи, щоб забезпечити ефективну роботу для користувачів та адміністраторів.

Основна мета цієї роботи полягає в розробці довідково-інформаційної системи для кінотеатру з реалізацією онлайн-продажу квитків. Головним завданням проекту є створення зручного та ефективного інструменту для користувачів, що дозволить їм швидко та легко знаходити необхідну інформацію про фільми, ознайомлюватись з розкладом сеансів, придбавати квитки безпосередньо через додаток.

При розробці системи було визначено кілька ключових цілей. Перш за все, метою було забезпечити зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко орієнтуватись у функціоналі додатку та з легкістю здійснювати необхідні дії. Другою ціллю було забезпечити повну та актуальну інформацію про фільми, включаючи опис, трейлери, рейтинги та відгуки, що допоможе користувачам зробити правильний вибір.

Крім того, основною метою проекту було реалізувати функціонал онлайн-продажу квитків зі зручними опціями оплати та інтеграцією з різними платіжними системами. Це дозволить користувачам придбати квитки на бажаний сеанс безпосередньо через додаток, ефективно уникнувши черг та затративши мінімум часу. В цілому, основна мета проекту полягає в розробці інноваційного та функціонального рішення, що покращить досвід відвідування кінотеатру та сприятиме розвитку онлайн

Нефункціональні вимоги: це вимоги, які стосуються якості та характеристик системи. Наприклад, надійність, швидкодія, безпека, зручність інтерфейсу користувача, масштабованість та сумісність з різними пристроями.

Вимоги до даних: це вимоги, пов'язані з обробкою, збереженням та захистом даних в системі. Наприклад, необхідність зберігання інформації про фільми, сеанси, користувачів, покупки квитків тощо. Також важливо визначити вимоги щодо захисту особистих даних користувачів та механізмів шифрування.

Вимоги до інтерфейсу користувача: Це вимоги, пов'язані з зовнішнім виглядом та взаємодією з користувачем. Наприклад, зручність інтерфейсу, зрозумілість навігації, наявність пошукової функції, можливість відображення постерів фільмів тощо.

Вимоги до безпеки: це вимоги, пов'язані з захистом системи від несанкціонованого доступу, злому або витоку даних. Наприклад, система повинна мати механізми аутентифікації та авторизації користувачів, шифрування чутливих даних, резервне копіювання та відновлення інформації.

Вимоги до масштабованості: Це вимоги, пов'язані з можливістю розширення та масштабування системи у разі зростання обсягу даних або кількості користувачів. Наприклад, система повинна бути готовою до обробки великої кількості запитів одночасно та масштабуватися шляхом додавання нових серверів або ресурсів.

Вимоги до сумісності: Це вимоги, пов'язані зі здатністю системи працювати з іншими програмними засобами, пристроями або стандартами. Наприклад, система повинна бути сумісною з різними операційними системами, веб-браузерами, мобільними пристроями або стандартами обміну даними.

Вимоги до підтримки та обслуговування: це вимоги, пов'язані з легкістю підтримки, обслуговування та оновлення системи. Наприклад, система повинна мати зручний інтерфейс для адміністраторів, механізми резервного копіювання та відновлення даних, логування подій та відстеження помилок.

Технічне завдання на розробку подану у додатку А.

1.4 Висновок. Постановка задачі

У результаті дослідження предметної області та постановки задачі було виявлено, що існуючі додатки для продажу квитків на сеанси у кінотеатрі мають деякі обмеження та недоліки. Розробка нової довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків на базі Vue.js з використанням API Vue Router та Vuex є актуальною та перспективною задачею, яка дозволить поліпшити взаємодію між кінотеатром та його клієнтами, забезпечить більш зручний та швидкий доступ до інформації про розклад сеансів та можливість купівлі квитків, а також забезпечить ефективне керування контентом та моніторинг продажів для адміністраторів кінотеатру

Проведене дослідження також дозволило зрозуміти, що використання технологій Vue.js, Vue Router та Vuex є дуже ефективним для розробки веб-додатків з клієнтською частиною. Ці технології дозволяють розробникам створювати високоякісні та динамічні інтерфейси, сприяють реактивному програмуванню та забезпечують зручну організацію даних та стану додатку.

Отже, мета розробки довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків має явний практичний та соціальний сенс, оскільки покращить якість обслуговування клієнтів кінотеатру, забезпечить швидкий доступ до необхідної інформації та покупки квитків, а також дозволить керувати контентом та моніторити продажі з боку адміністраторів.

Були проведені дослідження та розроблено довідково-інформаційну систему для кінотеатру з онлайн-продажем квитків. Аналіз існуючих додатків дав змогу виявити їхні недоліки та проблеми, що надали основу для створення нового функціонального рішення. Основна мета проекту полягала в розробці зручного та ефективного інструменту для користувачів, який дозволяє легко знаходити інформацію про фільми, переглядати розклад сеансів, реєструватися та купувати квитки онлайн.

Аналіз існуючих додатків показав, що багато з них мають недостатню інтеграцію з платіжними системами або обмежені можливості оплати. У цьому проекті було намічено мету забезпечити широкі можливості для онлайн-продажу квитків, включаючи підтримку різних платіжних систем та варіантів оплати. Це сприятиме зручності користувачів та забезпечить безпечну та ефективну процедуру придбання квитків через додаток.

У рамках проекту було визначено, що існуючі додатки для кінотеатрів не завжди надають достатньо повної та актуальної інформації про фільми, що може ускладнювати вибір для користувачів. Тому одним з головних завдань проекту є забезпечення повної та актуальної інформації про фільми для полегшення вибору користувачів.

В рамках розробки довідково-інформаційної системи для кінотеатру буде зроблено акцент на зборі та представленні різноманітної інформації про фільми.

Однак, завдяки проведеному дослідженню, була встановлена основна мета проекту - створити довідково-інформаційну систему, яка буде забезпечувати повну та актуальну інформацію про фільми, включаючи опис, трейлери, рейтинги та відгуки. Це допоможе користувачам зробити обґрунтований вибір та покращить загальний досвід перегляду фільмів у кінотеатрі.

Для досягнення цієї мети було здійснено вибір підходящих інструментів для розробки, зокрема Vue.js, який забезпечує ефективний та масштабований фронтенд розробку, а також Vuex для управління станом додатку. Також були використані Axios для взаємодії з API та Jest для розробки тестів.

На основі проведеного дослідження та використання відповідних інструментів, можна стверджувати, що розробка довідково-інформаційної системи для кінотеатру буде досягати своєї мети забезпечення якісної та доступної інформації для користувачів. Дана система допоможе покращити взаємодію між кінотеатром та його відвідувачами.

Після проведення дослідження предметної області та виявлення недоліків існуючих додатків для кінотеатрів, було сформульовано основну мету проекту - створення довідково-інформаційної системи для кінотеатру з реалізацією

онлайн-продажу квитків. Ця система має на меті надати користувачам повну та актуальну інформацію про фільми, що дозволить їм зробити обґрунтовані вибори та забезпечить зручний та задоволений досвід перегляду фільмів.

Для реалізації проекту було використано ряд технологій та інструментів. Фронтенд частина була побудована з використанням Vue.js, що забезпечило швидку та ефективну розробку інтерфейсу користувача. Для керування станом додатку був використаний Vuex, що спрощує управління даними та їх синхронізацію між компонентами.

Також в проекті було використано Axios для здійснення HTTP-запитів до сервера та отримання необхідних даних, а Jest був використаний для написання тестів та перевірки правильності роботи різних функцій та компонентів. Ці інструменти допомогли забезпечити якість та надійність системи.

У результаті дослідження предметної області, постановки задачі та вибору відповідних технологій було розроблено довідково-інформаційну систему для кінотеатру з реалізацією онлайн-продажу квитків. Ця система вирішує проблему обмеженої та неповної інформації про фільми, що існує у багатьох існуючих додатках. Вона надає користувачам зручний спосіб знайти необхідну інформацію про фільми, ознайомитися з їх описами та трейлерами, переглянути рейтинги та відгуки.

Постановка задачі для розробки довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків полягає у визначенні конкретних цілей та функцій, які система повинна виконувати. Основною метою постановки задачі є створення зручного та ефективного інструменту для кінотеатру, що дозволить клієнтам отримувати доступ до інформації про фільми та здійснювати онлайн-продаж квитків без зайвих зусиль.

Основні задачі, що вирішуються у рамках проекту:

– розробка інтерфейсу: створення зручного та привабливого веб-інтерфейсу, який дозволить користувачам легко навігувати сайтом та знайти необхідну інформацію про фільми, графік сеансів та купити квитки; інтерфейс

має бути інтуїтивно зрозумілим та мобільно-дружнім, щоб забезпечити зручний доступ до системи з різних пристроїв;

– управління фільмами та сеансами: реалізація функціоналу для додавання та редагування інформації про фільми, включаючи назву, опис, тривалість та постер; також система повинна забезпечувати можливість створення розкладу сеансів, визначення цін на квитки та керування доступністю квитків для продажу.

– онлайн-продаж квитків: розробка механізму, що дозволяє користувачам обирати фільм, сеанс та місце в залі, а потім здійснювати онлайн-платіж для придбання квитків; система повинна забезпечувати вибір різних типів квитків

– керування користувачами та облік даних: система повинна мати можливість реєстрації нових користувачів, аутентифікації і авторизації вже зареєстрованих користувачів; крім того, необхідно забезпечити збереження особистої інформації користувачів, включаючи дані про платежі, контактну інформацію та історію покупок; важливо використовувати механізми шифрування та забезпечити конфіденційність цих даних.

– звітність та аналітика: система повинна забезпечувати можливість створення звітів та аналітичних даних, що дозволяють керівництву кінотеатру отримувати інформацію про продажі, популярність фільмів, найбільш вигідні сеанси та іншу аналітику, необхідну для прийняття ефективних рішень щодо розкладу сеансів та маркетингових акцій.

– масштабованість та надійність: система повинна бути готовою до масштабування, тобто здатною обробляти великий обсяг користувацьких запитів та транзакцій; необхідно планувати архітектуру системи таким чином, щоб вона була готовою до розширення та збільшення продуктивності з ростом користувацької бази та обсягу операцій.

В цілому, постановка задачі полягає в розробці комплексної довідково-інформаційної системи для кінотеатру, яка забезпечує зручну онлайн-покупку квитків та надійний доступ до інформації про фільми та сеанси. Враховуючи всі вимоги та функціональні можливості системи. Можливість додавання,

редагування та видалення інформації про фільми, включаючи назву, опис, тривалість та постер. Керування розкладом сеансів, визначенням цін на квитки та доступністю квитків для продажу. Реалізація механізму онлайн-продажу квитків, що дозволяє користувачам обирати фільм, сеанс та місце в залі, а потім здійснювати онлайн-платіж для придбання квитків.

Основна мета полягає у забезпеченні доступу до актуальної та повної інформації про фільми, графік сеансів та квитки для користувачів. Для цього система повинна мати наступні функції та характеристики

Інформаційна база даних: Розроблена система повинна мати добре структуровану базу даних, яка містить повну інформацію про фільми, включаючи назву, жанр, тривалість, режисера, акторський склад та опис. Також потрібно мати інформацію про графік сеансів, включаючи час та дату, доступність квитків та залу.

2 ПРОЕКТУВАННЯ ДОВІДКОВО-ІНФОРМАЦІЙНОЇ СТСТЕМИ

2.1 Проектування архітектури та структури системи.

Архітектура клієнт-сервер є популярним розподіленою архітектурою, що використовується для побудови програмного забезпечення. Вона заснована на концепції розподіленого обчислення, де система розбивається на дві основні компоненти: клієнтів і серверів, які взаємодіють між собою через мережу, можна графічно зобразити цю архітектуру (рисунок 2.1).

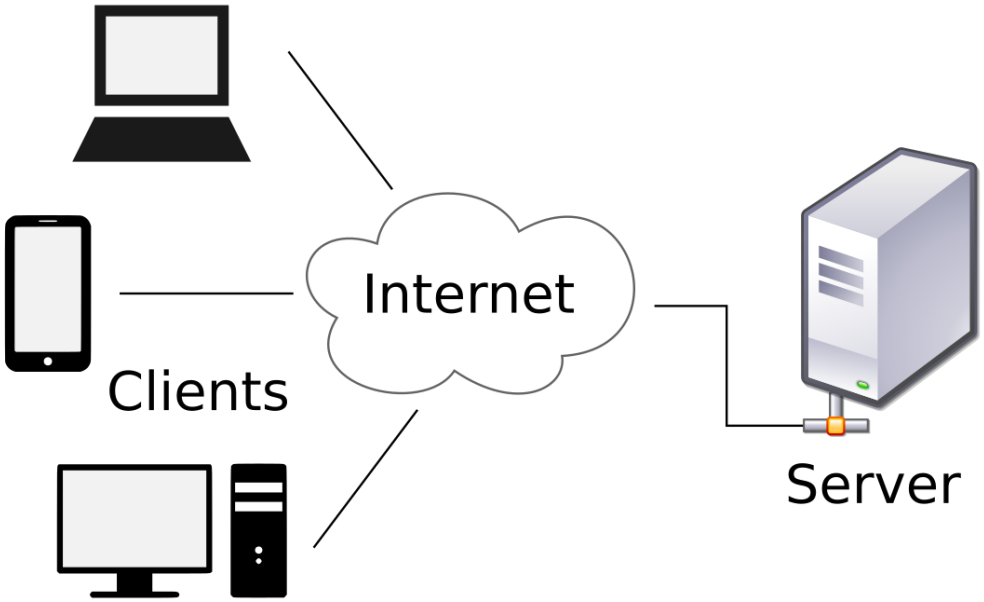


Рисунок 2.1 – Архітектура клієнт-сервер [3]

Клієнт – це користувачський інтерфейс або додаток, який надає можливість взаємодії з користувачем. Клієнтська частина може бути реалізована у вигляді десктопної програми, веб-браузера, мобільного додатка або будь-якого іншого засобу, що дозволяє користувачеві взаємодіяти з системою. Клієнтська частина зазвичай надсилає запити до сервера і обробляє отримані відповіді, відображаючи їх для користувача.

Сервер – це централізований компонент, який надає певні послуги клієнтам. Він може бути фізичним комп'ютером або кластером комп'ютерів, що забезпечують ресурси та функціональні можливості. Сервер обробляє запити, надсилані клієнтами, виконує необхідні операції та надсилає відповіді назад клієнтам. Він також може зберігати та керувати даними, виконувати бізнес-логіку і забезпечувати доступ до різноманітних ресурсів.

Клієнти та сервери взаємодіють за допомогою мережі, такої як Інтернет або локальна мережа. Клієнтська частина надсилає запити до сервера, передаючи необхідну інформацію, таку як параметри запиту або дані. Сервер обробляє запити та повертає відповідь з результатами запиту клієнту. Цей процес відбувається через встановлену комунікаційну протоколу, наприклад, HTTP, TCP/IP або інший протокол, який використовується для передачі даних між клієнтом та сервером.

Структура проекту "Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків" може бути описана наступним чином:

– config/: папка з файлами конфігурації проекту, включаючи налаштування для підключення до API кінотеатру та налаштування маршрутизації;

– public/: папка, яка містить статичні файли, які будуть доступні для відображення на сторінці, такі як зображення, стилі та скрипти;

– src/: це головна папка проекту, яка містить всі вихідні файли проекту;

Папка src/ складається з наступних підпапок:

– api/: це папка, яка містить файли для підключення до API кінотеатру за допомогою Axios. Вона містить такі файли, як auth.js, cinema.js, movies.js тощо.

– assets/: це папка, яка містить всі ресурси, необхідні для проекту, такі як зображення, шрифти та іконки;

– components/: це папка, яка містить всі Vue-компоненти, які використовуються для створення інтерфейсу користувача. Вона поділена на підпапки згідно з їх функціональним призначенням, такі як "layouts", "modals", "movies", "cinema" тощо;

– router/: це папка, яка містить налаштування маршрутизації додатку з використанням Vue Router;

– store/: це папка, яка містить всі файли для зберігання даних за допомогою Vuex, які використовуються для управління станом додатку;

– utils/: це папка, яка містить утиліти та допоміжні функції, які використовуються в проекті;

– views/: це папка, яка містить всі Vue-компоненти, які представляють сторінки додатку;

– App.vue: це головний компонент додатку, який містить всі інші компоненти та відобража.

На рисунку 2.2 представлена файлова структура проекту.

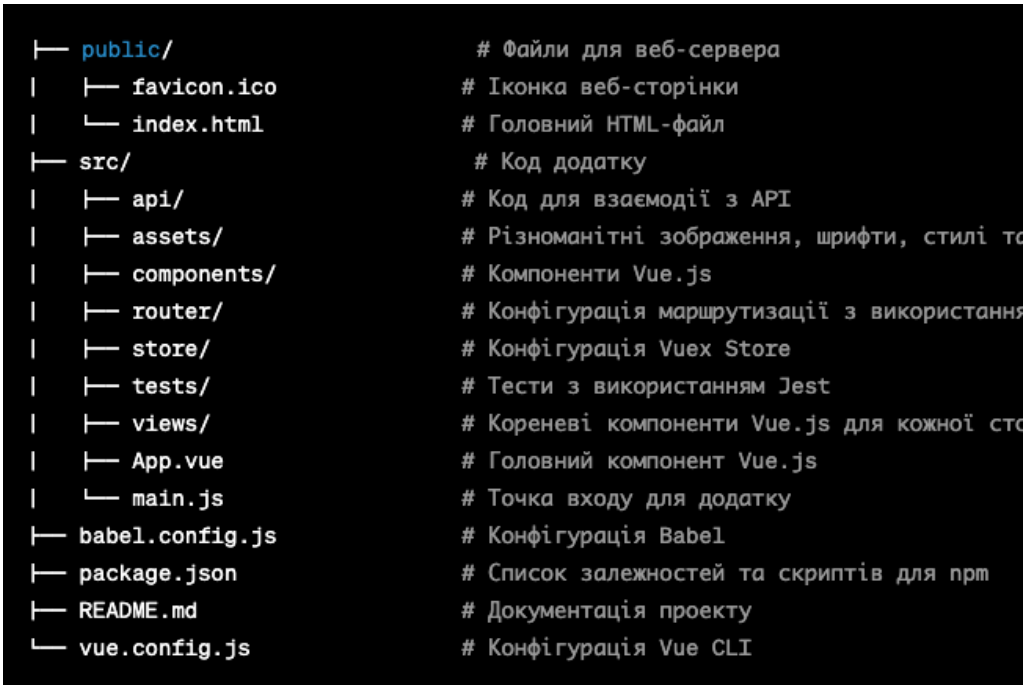


Рисунок 2.2 – Файлова структура

На рисунку 2.2 зображена типова структура проекту Vue.js з декількома папками для коду додатку (src), файлами конфігурації (babel.config.js, package.json, README.md, vue.config.js) та папкою для веб-сервера (public). У папці src знаходяться різні підпапки для компонентів Vue.js, конфігурації маршрутизації, Vuex Store, API та іншого коду, необхідного для розробки

проекту. Також у папці tests знаходяться тести, написані з використанням Jest. Кожен компонент Vue.js розташований у своїй власній папці та містить шаблон, логіку та стилі компонента. Основний файл додатку – це main.js, а головний компонент – це App.vue.

Перш за все, варто розглянути підхід до організації файлів та каталогів. Рекомендовано використовувати підхід "розділена за функціональністю" або "розділена за фічами", де кожна функціональна одиниця проекту розміщується у відповідній папці. Це дозволяє легко знайти та розуміти структуру проекту, особливо при зростанні його розміру.

Другий аспект – це організація компонентів та модулів. Використання модульної структури дозволяє розбити проект на незалежні компоненти, що полегшує їх розробку, тестування та підтримку. В рамках Vue.js можна використовувати концепцію одного компонента на файл або групувати компоненти в підкаталоги в залежності від їхньої функціональності.

Третій аспект – це розробка модульної структури сторінок. Якщо ваш проект має багато сторінок, корисно розділити їх на окремі модулі або компоненти, що представляють конкретні сторінки. Це допоможе зберігати код сторінки в одному місці та робити його більш управляємим.

Четвертий аспект – це розробка загальної структури даних. Використання станових управлінь, таких як Vuex для Vue.js, дозволяє зберігати та керувати станом даних у вашому додатку.

Загальна структура проекту також може включати директорії та файли для обробки роутінгу та навігації. Для цього ви можете використовувати бібліотеку роутингу, таку як Vue Router, яка дозволяє визначити маршрути та пов'язані з ними компоненти. Це дозволяє керувати навігацією у вашому додатку та переключатися між різними сторінками чи компонентами.

При розробці структури проекту також слід враховувати можливість розширення та підтримки. Використання модульної структури дозволяє легко додавати нові функціональності, компоненти чи модулі до проекту без великих змін усього коду. Це особливо корисно, коли розробка проекту відбувається у

команді, де кожен член може працювати над окремим модулем або компонентом. На базі цього було зроблені креслення діаграм, а саме UML-діаграма класів (рисунок 2.3).

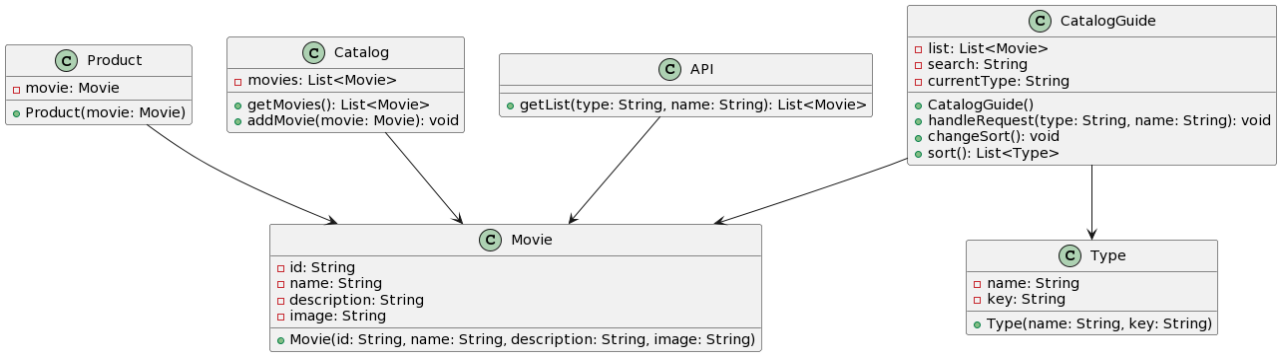


Рисунок 2.3 – UML-діаграма класів

Також була зроблена UML-діаграма послідовності, яка описує послідовність дій та взаємодію між різними компонентами системи під час процесу продажу квитків онлайн. Клієнт обирає фільм та кількість квитків, які він хоче придбати, і передає цю інформацію фронтенду (рисунок 2.4).

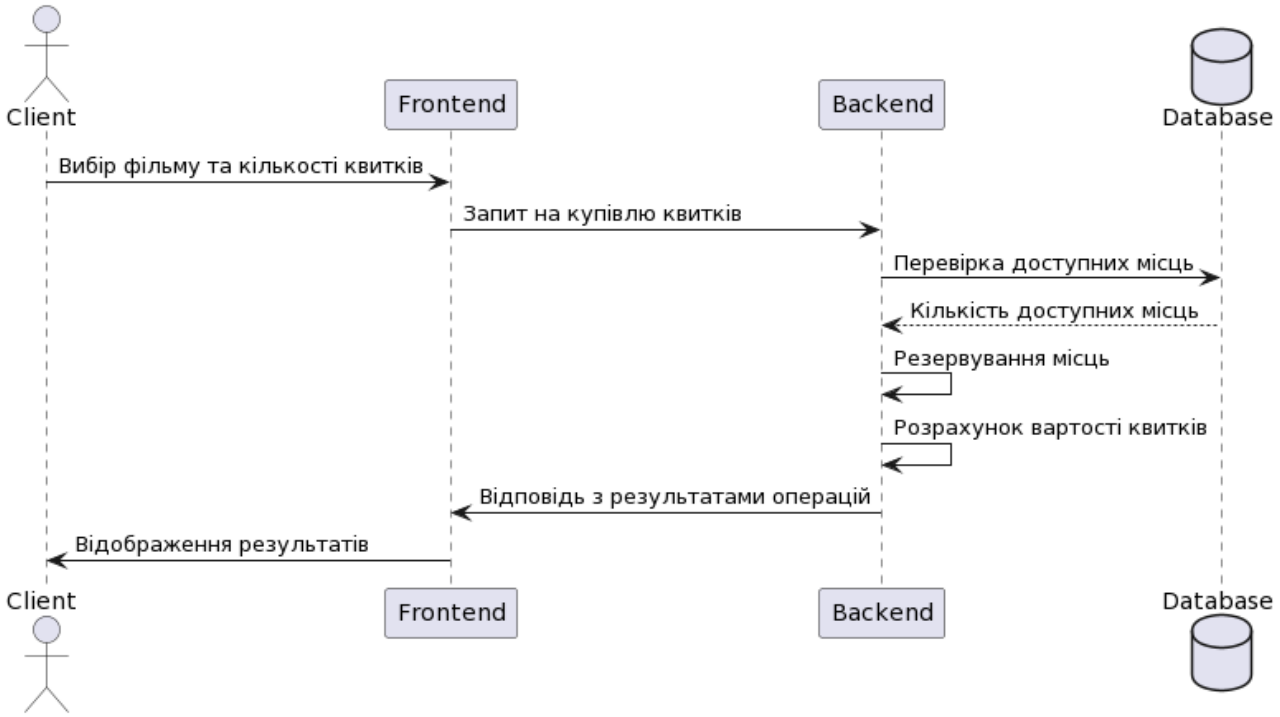


Рисунок 2.4 – UML-діаграма послідовності

Також для простішого представлення, що має робити програма, була створена діаграма діяльності. У цій діаграмі діяльності показано послідовність кроків, які користувач повинен виконати для купівлі квитків у системі онлайн-продажу квитків (рисунок 2.5).

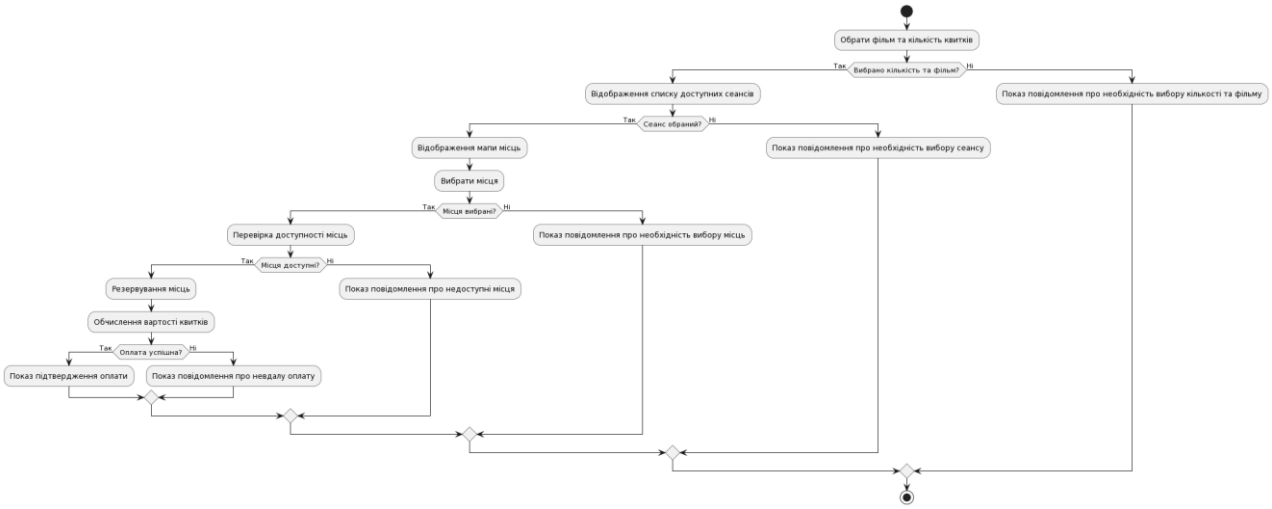


Рисунок 2.5 – UML-діаграма діяльності

Для полегшення розробки та підтримки також рекомендується використовувати правила та патерни неймінгу файлів та компонентів. Це допомагає зрозуміти, які файли відповідають за які компоненти, і спрощує пошук та редагування коду. Крім того, варто дотримуватися стандартів та керуватися рекомендаціями спільноти, щоб забезпечити читабельність та однорідність коду у всьому проекті.

Не менш важливим аспектом структури проекту є належна документація. Розробка описів компонентів, модулів та їх функціональності, а також документування патернів та правил, що використовуються в проекті, допомагає команді розробників краще розуміти код та спрощує спілкування та співпрацю.

2.2 Проектування бази даних

Проектування бази даних є важливим етапом у розробці довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків. Цей етап включає створення оптимальної структури бази даних, яка забезпечить ефективне зберігання та управління необхідною інформацією.

Першим кроком у проектуванні бази даних є аналіз вимог та потреб системи. На цьому етапі встановлюються сутності (такі як фільми, кінотеатри, клієнти, сеанси тощо), а також взаємозв'язки між ними. Важливо врахувати всі потрібні атрибути для кожної сутності та визначити їх типи даних для цього була створено модель бази даних (рисунок 2.6).

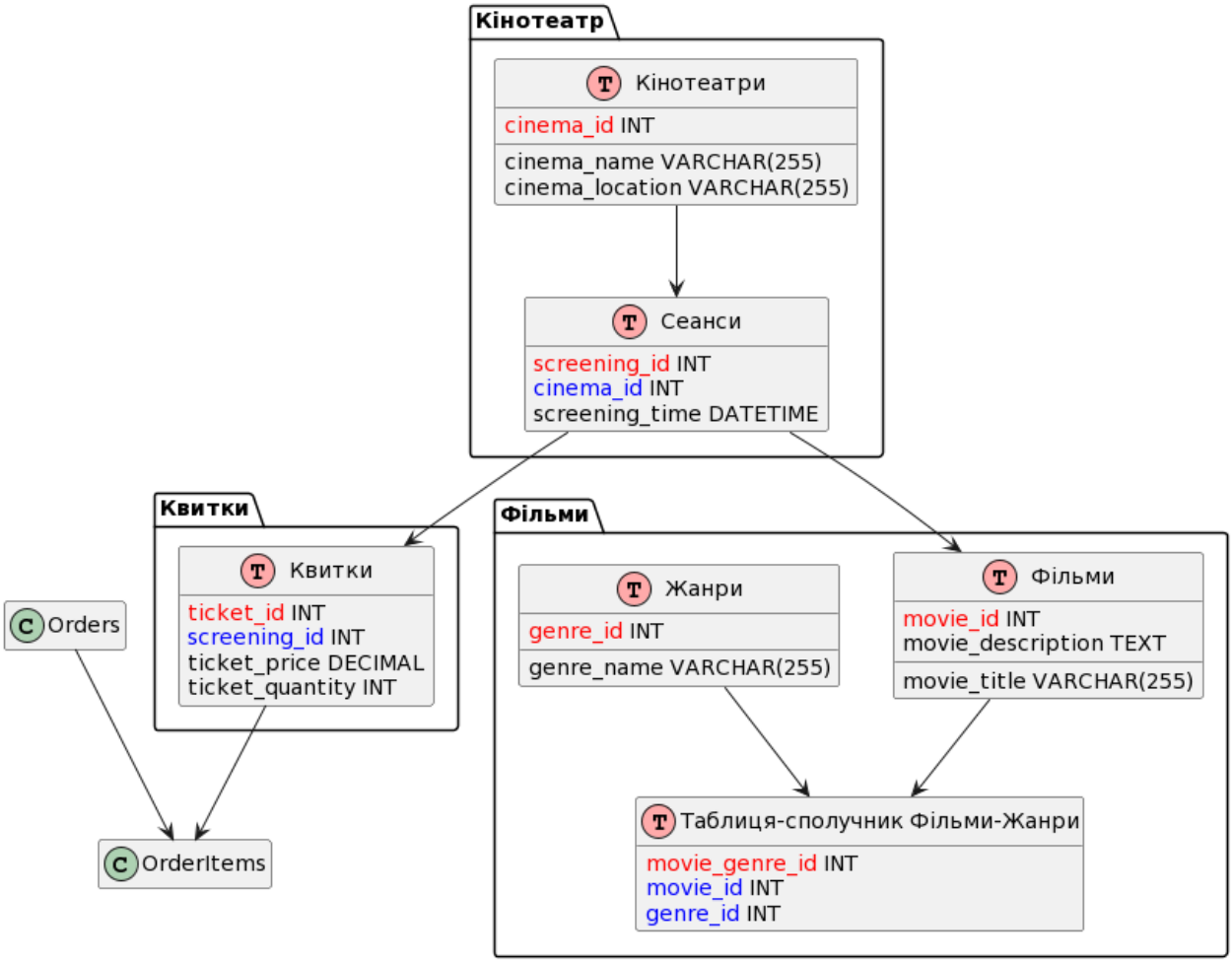


Рисунок 2.6 – Модель бази даних

Далі проводиться нормалізація бази даних. Цей процес допомагає уникнути дублювання даних та забезпечити їх цілісність і консистентність. Нормалізація включає розбиття таблиць на менші логічні сутності та встановлення зв'язків між ними за допомогою первинних та зовнішніх ключів.

Після нормалізації розробляються таблиці бази даних з визначенням потрібних полів для кожної таблиці. Для кожного поля встановлюється його тип даних, обмеження, а також індекси для покращення швидкодії пошуку, сортування та фільтрації.

Також у проектуванні бази даних враховуються запити, які будуть виконуватись над базою даних. Це допомагає оптимізувати структуру бази даних та вибрати підходящі індекси для швидкого виконання запитів.

Завершальним кроком є розробка схеми бази даних, яка включає усі таблиці, їх поля, зв'язки між таблицями та ключем. Ця схема стає основою для подальшої реалізації бази даних та її інтеграцією.

Проектування бази даних вимагає уважного аналізу потреб системи та визначення її структурних особливостей. Цей етап є ключовим для успішної реалізації довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків, оскільки правильно спроектована база даних забезпечить ефективну та надійну роботу системи.

2.3 Проектування інтерфейсу користувача.

Проектування інтерфейсу користувача є важливою складовою довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків. Цей пункт передбачає розробку зручного та інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам з легкістю взаємодіяти з системою та здійснювати необхідні операції для цього була розроблена модель інтерфейсу яка відображає що потрібно буде мати в цьому додатку (рисунок 2.7).

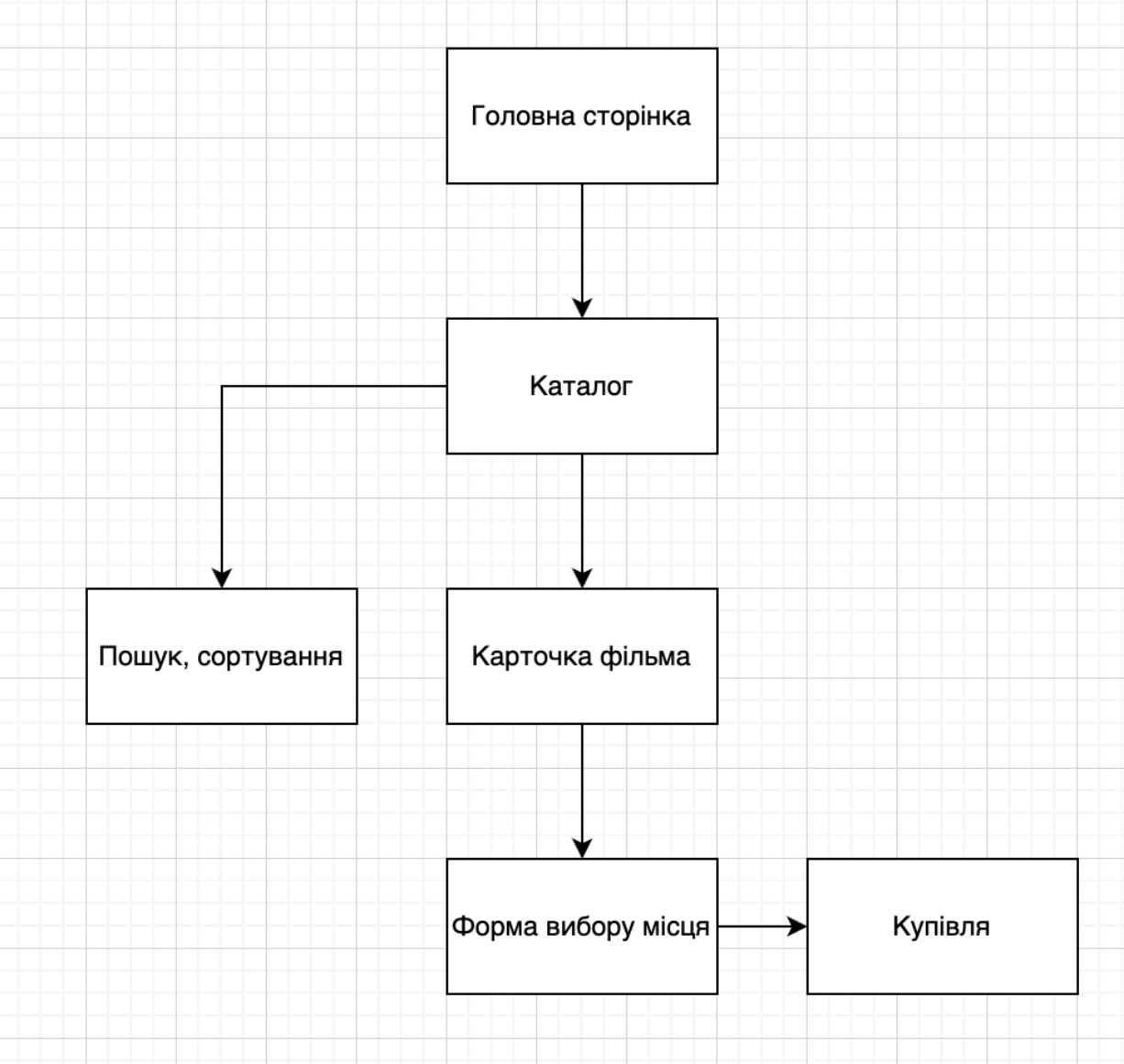


Рисунок 2.7 – Модель інтерфейсу користувача

Першим кроком у проектуванні інтерфейсу є аналіз потреб користувачів та їх вимог щодо функціональності системи. Важливо зрозуміти, які завдання та операції користувачі будуть виконувати, і яким чином система повинна їх підтримувати. На цьому етапі визначаються основні функції та функціональні блоки системи.

Далі проводиться проектування візуальної складової інтерфейсу, таких як макети, кольорова схема, типографіка тощо. Важливо створити зручну та привабливу графічну оболонку, яка буде привертати увагу користувачів та спонукає їх до взаємодії з системою.

При проектуванні інтерфейсу також враховуються принципи юзабіліті та доступності. Інтерфейс повинен бути легким у використанні, інтуїтивно зрозумілим та ефективним. Користувачі повинні швидко знаходити необхідну інформацію та легко виконувати операції без зайвого зусилля.

Крім того, розробка інтерфейсу включає визначення навігації та організацію контенту. Структура інформації повинна бути логічною та ієрархічною, зручною для переходу між різними розділами та функціями.

У процесі проектування інтерфейсу важливо проводити тестування та збирати фідбек від користувачів, щоб виявити та виправити можливі проблеми та недоліки. Застосування найкращих практик та стандартів дизайну допоможе створити зручний та привабливий інтерфейс, що задовольнятиме потреби користувачів та сприятиме ефективній роботі з довідково-інформаційною системою кінотеатру.

В цілому, проектування інтерфейсу користувача передбачає розробку зручного та привабливого інтерфейсу, що відповідає потребам користувачів та сприяє ефективній роботі з системою. Цей процес включає аналіз потреб користувачів, визначення функціональних блоків та візуального оформлення інтерфейсу. Результатом проектування є зручний, інтуїтивно зрозумілий та привабливий інтерфейс, який забезпечує комфортну взаємодію користувачів з довідково-інформаційною системою кінотеатру.

2.4 Аналіз та вибір технологій і методів реалізації системи

Після проведення аналізу вимог та предметної області, було вирішено використовувати технології Vue.js, Vue Router та Vuex для розробки довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків. Vue.js було вибрано через його зручну та просту структуру, широкі можливості для розробки клієнтської частини, високу продуктивність та масштабованість. Vue

Router дозволяє зручно налаштовувати маршрутизацію та навігацію в додатку, а Vuex – забезпечує ефективне управління станом додатку та реалізацію зв'язку між компонентами

Vue.js є прогресивним JavaScript-фреймворком, який використовується для побудови користувацького інтерфейсу веб-додатків. Одним з головних переваг Vue.js є його простота використання, що робить його доступним для розробників з різним рівнем досвіду. Vue.js пропонує компонентну архітектуру, де кожен елемент інтерфейсу представляє собою окремий компонент, що спрощує розподіл роботи та робить код більш організованим.

Однією з головних особливостей Vue.js є реактивність. Завдяки використанню реактивних об'єктів та директив, Vue.js автоматично відслідковує зміни в даних та оновлює відповідні елементи інтерфейсу. Це дозволяє створювати живі та динамічні додатки, де зміни відображаються миттєво без необхідності вручну оновлювати стан інтерфейсу.

Vue.js також пропонує широкий спектр функціональних можливостей. Він має вбудовану систему маршрутизації, що дозволяє легко налаштувати навігацію в додатку. Крім того, Vue.js надає можливість використовувати розширені директиви, фільтри та механізми валідації для забезпечення якості та потужності розроблюваного додатку.

Окрім цього, Vue.js має активну та підтримувану спільноту розробників, що сприяє обміну знаннями та розвитку екосистеми інструментів. Значна кількість документації, пакетів розширень та статей дозволяє розробникам ефективно використовувати Vue.js та швидко вирішувати завдання

Vue.js та Vuex є взаємодоповнюючими інструментами, які допомагають розробникам побудувати потужні та масштабовані веб-додатки. Vue.js забезпечує реактивну та компонентну архітектуру, тоді як Vuex є офіційним становим менеджером для Vue.js, допомагаючи керувати складним станом додатку та спрощуючи його синхронізацію між компонентами.

Vuex є централізованим сховищем даних для Vue.js додатків. Він зберігає усі стани додатку в одному місці, що робить управління та маніпулювання

станом додатку простим та прозорим. За допомогою Vuex, розробники можуть встановлювати правила для зміни стану та забезпечувати, щоб зміни були мутаціями, що зроблює код більш прогнозованим та легким для налагодження.

Одним з ключових концепцій Vuex є використання строгого однонаправленого потоку даних. Це означає, що стан додатку може бути змінений тільки за допомогою здійснення мутацій, а доступ до стану здійснюється через геттери. Це підвищує прогнозованість та зрозумілість коду, а також полегшує відлагодження та тестування додатку.

З використанням Vuex, розробники можуть ефективно управляти станом додатку, розділяючи його на модулі. Це дозволяє групувати стосовно пов'язані дані та логіку в окремі модулі, що полегшує масштабування додатку та зберігає його організованим. Також, Vuex надає можливість використовувати серединні програми (middlewares) для обробки асинхронних операцій.

Vue.js є одним з найпопулярніших фреймворків для розробки веб-додатків, і він демонструє кілька переваг у контексті довідково-інформаційної системи кінотеатру. Перш за все, Vue.js має простий та інтуїтивно зрозумілий синтаксис, що полегшує розробку та підтримку коду. Це особливо важливо в проекті, який має швидкий термін впровадження і вимагає частої зміни та доповнення функціональності.

Другим важливим аспектом є легка інтеграція Vue.js з іншими бібліотеками та плагінами. У контексті довідково-інформаційної системи кінотеатру, де можуть використовуватися різноманітні компоненти, бібліотеки та інструменти, зручна інтеграція є важливою перевагою. Vue.js пропонує гнучкість у виборі технологій та дозволяє легко впроваджувати рішення.

Також, Vue.js має велику активну спільноту розробників, яка забезпечує широкий спектр документації, плагінів, компонентів та інших ресурсів. Це дозволяє ефективно вирішувати проблеми та знаходити рішення шляхом спілкування зі спільнотою та використанням готових рішень. Крім того, активна спільнота сприяє постійному розвитку та покращенню фреймворку, забезпечуючи нові функції та виправлення помилок.

Таким чином, вибір Vue.js для розробки довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків є обґрунтованим рішенням, оскільки цей фреймворк пропонує простоту використання, гнучкість у виборі технологій та підтримку активної спільноти розробників. В результаті, розробка додатку на Vue.js сприятиме швидкому впровадженню, забезпеченню високої якості та покращенню користувацького досвіду.

Інтеграція Nuxt.js у довідково-інформаційну систему кінотеатру може принести кілька переваг та покращень. Ось кілька способів, які можна використати для покращення додатку за допомогою Nuxt.js. Покращена продуктивність: Nuxt.js надає можливість побудувати універсальний додаток. Це означає, що сторінки будуть попередньо рендеритися на сервері, що поліпшує швидкість завантаження та відкликання. Користувачі будуть бачити швидшу ініціалізацію додатку та загально кращий досвід використання.

Управління маршрутизацією: Nuxt.js надає розширені можливості для управління маршрутизацією в додатку. За допомогою Nuxt.js можна легко налаштувати динамічні маршрути, статичну генерацію сторінок та інші функції, що поліпшують навігацію та взаємодію користувача з додатком.

Збільшена масштабованість: Nuxt.js дозволяє ефективно масштабувати додаток за допомогою модульної структури та плагінів. Можна легко додавати нові функції та розширювати функціональність, що дає можливість зручно пристосовувати додаток до зростаючих потреб.

Збільшена безпека: Nuxt.js дозволяє використовувати різні методи захисту, такі як middleware та роутинг з автентифікацією, що підвищує рівень безпеки додатку. Можна налаштувати різні рівні доступу та захистити конфіденційну інформацію.

Інтеграція Nuxt.js з довідково-інформаційною системою кінотеатру допоможе покращити продуктивність, масштабованість, безпеку та управління маршрутизацією. Це забезпечить більш зручний та ефективний досвід користувачів, а також поліпшить розробку та підтримку додатку.

Таким чином, вибір цих інструментів дозволить розробникам зосередитись на реалізації функціональності додатку, забезпечить високу продуктивність та масштабованість, а також забезпечить зручне та швидке обслуговування користувачів довідково-інформаційної системи кінотеатру.

До вибору інструментів для розробки, було проведено аналіз можливостей різних шаблонізаторів, і в результаті було вирішено використовувати pug для створення розмітки.

Pug – це шаблонізатор, що дозволяє зменшити кількість написаного коду та полегшити читання HTML-коду. Замість звичайної розмітки HTML, Pug використовує зручну та лаконічну синтаксичну форму, що спрощує структурування HTML-сторінок та зменшує кількість написаного коду. Таким чином, використання. Pug дозволить значно зменшити кількість написаного коду, полегшити читання та розуміння HTML-коду, а також прискорити процес розробки та збірки проекту.

Один з головних принципів Pug – це значна економія часу та роботи завдяки його скороченому синтаксису. Замість використання тегів HTML, Pug використовує відступи та відповідний синтаксис, що дозволяє розробникам створювати розмітку швидше та без зайвих символів. Крім того, Pug надає можливість використовувати цикли, умовні оператори та функції для динамічної генерації розмітки.

Ще однією зручною функцією Pug є можливість вкладати елементи один в одного без необхідності використовувати закриваючі теги. Замість того, щоб вказувати відкриваючий та закриваючий теги, ви можете просто вкладати елементи один в одного за допомогою відступів, що робить код більш зрозумілим та легким для читання. Крім того, Pug дозволяє використовувати міксіни (mixins) для створення повторюваних блоків коду, що полегшує роботу зі стандартними компонентами.

Для здійснення запитів до сервера та отримання відповіді використовувався бібліотека Axios. Axios – це HTTP-клієнт для браузера та Node.js, що дозволяє легко здійснювати запити на сервер та отримувати

відповідь. Використання Axios в програмі дозволило зробити код більш читабельним та зменшити кількість коду, що було потрібно написати для здійснення запитів. Вона надає простий та зрозумілий інтерфейс для взаємодії з API та отримання даних з сервера. Axios підтримує всі основні методи HTTP, такі як GET, POST, PUT, DELETE, і дозволяє налаштовувати різні параметри запиту, такі як заголовки та параметри URL.

Одним з головних переваг Axios є його простота використання. Він надає проміс-подібний інтерфейс, що дозволяє використовувати ланцюжки обіцянок (promises) або асинхронні/очікувані функції (async/await) для зручного управління запитами та обробки їх результатів. Це дозволяє зменшити кількість коду та спростити асинхронну логіку.

Axios також надає можливість налаштування міжперехоплювачів (interceptors), що дозволяють перехоплювати та обробляти запити та відповіді перед їх відправкою або після отримання. Це дозволяє додавати заголовки, обробляти помилки, автоматично додавати аутентифікаційні токени та робити інші маніпуляції з запитами на глобальному рівні. Це робить Axios дуже гнучким та потужним інструментом для роботи з API.

Крім того, Axios підтримує обробку відповідей у різних форматах, таких як JSON, XML, Blob і т.д. Він також дозволяє завантажувати та відправляти файли за допомогою запитів, що дозволяє легко працювати з бінарними даними. Axios також надає можливість скасування запитів, що дозволяє уникнути надлишкового виконання запитів.

Jest – це фреймворк для тестування JavaScript, розроблений для платформ Node.js та React. Він надає зручний та простий інтерфейс для написання тестів та їх запуску. Основна перевага Jest полягає у тому, що він підтримує вбудовану інтеграцію з Babel та TypeScript, що дозволяє використовувати сучасний синтаксис JavaScript та сприяє зручному написанню тестів. Крім того, Jest має вбудовану підтримку відстеження змін (watch mode), що дозволяє автоматично виконувати тестування при кожній зміні коду, що забезпечує швидкий зворотний зв'язок та покращує ефективність розробки. Jest також надає

можливість виконувати різні види тестування, включаючи модульні, функціональні та інтеграційні тести, що дозволяє забезпечити повну покриття тестами та надійність коду.

Крім того, Jest підтримує виконання тестів в різних середовищах, включаючи Node.js, браузери та середовище розробки React Native. Застосування Jest у розробці програмного забезпечення дозволяє забезпечити високу якість коду, підвищити ефективність розробки та знизити кількість помилок у програмі.

Однією з ключових переваг Jest є його зручний та легкий у використанні синтаксис. Він надає чіткі та зрозумілі методи для опису тестових сценаріїв, таких як describe, it та expect. Це дозволяє розробникам швидко створювати та читати тести, спрощуючи процес тестування.

Jest також надає багато вбудованих функцій для мокування, що дозволяють замінювати реальні залежності на штучні, контрольовані значення під час виконання тестів. Це дозволяє ізолювати тести від залежностей та забезпечує більш точні та надійні результати. Jest також підтримує вбудовані функції для перехоплення та спостереження за функціями, що дозволяє перевіряти, чи викликалися певні функції з певними параметрами.

Для розробки довідково-інформаційної системи для кінотеатру з онлайн-продажем квитків було прийнято рішення використовувати декілька потужних інструментів та технологій.

Першим з них був вибір фейкового API. Використання фейкового API дозволило нам створити тестовий сервіс, який емулює взаємодію з реальним API, забезпечуючи нам зручне середовище для розробки та тестування функціоналу без прив'язки до реального сервера. Це спрощує і прискорює процес розробки та дозволяє перевірити різні сценарії роботи додатку.

Другим важливим інструментом було використання Vue.js – прогресивного JavaScript-фреймворку для побудови користувацького інтерфейсу. Vue.js надає зручну та ефективну структуру для розробки складних

веб-додатків, а також пропонує багато корисних функцій, таких як компонентна архітектура, реактивність та простота використання.

Для керування станом додатку було використано Vuex – офіційний становий менеджер для Vue.js. Vuex забезпечує централізоване керування станом додатку, дозволяючи зручно управляти даними та забезпечити їх синхронізацію між різними компонентами додатку. Це спрощує управління складними станами та забезпечує консистентність даних у всьому додатку.

2.5 Висновок

Належна структура проекту є важливим аспектом розробки програмного забезпечення. Вона допомагає забезпечити організовану та легкозмінну архітектуру, що сприяє швидкій розробці, підтримці та розширенню проекту. Використання правильних інструментів, таких як Vue.js, Vuex, Axios та Jest, допомагає забезпечити ефективну розробку та тестування проекту.

Vue.js є потужним фреймворком JavaScript, який надає зручність та простоту у розробці користувацького інтерфейсу. Використання Vue.js дозволяє створювати компонентну архітектуру, яка полегшує розбиття проекту на незалежні компоненти та сприяє повторному використанню коду.

Vuex, як становий управлінь для Vue.js, забезпечує централізоване зберігання та керування станом даних у додатку. Використання Vuex дозволяє забезпечити консистентність та доступність даних у всьому додатку, спрощуючи керування станом та спільну роботу з даними.

Axios є потужною бібліотекою для виконання HTTP-запитів у JavaScript. Його використання дозволяє взаємодіяти з сервером та отримувати чи надсилати дані асинхронно. Axios забезпечує зручний інтерфейс та можливості для обробки помилок, керування заголовками та інші функціональності, що полегшують роботу з мережевими запитами.

Jest – це потужний фреймворк тестування, що спеціально розроблений для тестування JavaScript-коду. Використання Jest дозволяє писати й запускати автоматизовані тести для перевірки правильності роботи вашого коду. Jest також має вбудовану підтримку для снапшот-тестування, що дозволяє зберігати "знімки" (snapshots) результатів тестів і порівнювати їх з майбутніми викликами, щоб впевнитися, що зміни в коді не вплинули на очікувані результати. Це спрощує перевірку відповідності результатів під час рефакторингу або змін в коді.

Загальна структура проекту, яка використовує Vue.js, Vuex, Axios та Jest, дозволяє створити добре організовану та підтримувану архітектуру для вашого додатку. Використання модульної структури для компонентів, модулів та сторінок дозволяє зберігати код окремо та забезпечує легку розширюваність та підтримку. Також використання відповідних інструментів, таких як Axios для мережевих запитів та Jest для автоматизованого тестування, сприяє надійності та якості вашого додатку.

Необхідно також враховувати індивідуальні потреби вашого проекту при виборі та організації структури та технологій. Розуміння та використання кращих практик у розробці, таких як розділення за функціоналом, модульність та документація, допомагає створити масштабований та легкозмінний проект. Загальна структура проекту разом з використанням Vue.js, Vuex, Axios та Jest може допомогти забезпечити успішну та продуктивну розробку вашого додатку.

Крім того, використання Vue.js, Vuex, Axios та Jest сприяє збільшенню продуктивності розробників. Vue.js забезпечує простоту та ефективність у створенні компонентів та їх взаємодії, що дозволяє розробникам швидко створювати функціональність та вигляд додатку. Vuex забезпечує централізоване керування станом, що дозволяє розробникам легко взаємодіяти зі станом додатку та забезпечує консистентність даних.

Axios є потужним інструментом для взаємодії з сервером, надаючи простий інтерфейс для виконання HTTP-запитів. Його використання спрощує комунікацію з сервером та обробку отриманих даних. Бібліотека також надає

можливість керувати заголовками, обробляти помилки та працювати з різними типами даних.

Jest, як потужний фреймворк для тестування, надає зручний спосіб писати, запускати та управляти автоматизованими тестами. Він дозволяє розробникам перевіряти правильність роботи коду та покривати його тестами, що забезпечує більшу впевненість у функціональності та надійності додатку. Jest також має широкі можливості для тестування, включаючи снапшот-тестування, що полегшує перевірку відповідності результатів тестів з очікуваними значеннями.

Загалом, використання Vue.js, Vuex, Axios та Jest допомагає створити потужний, ефективний та добре структурований проект. Ці технології забезпечують зручну розробку користувацького інтерфейсу, ефективну роботу з даними та взаємодію з сервером.

Одним з ключових виборів було використання фреймворку Vue.js для розробки фронтенд-частини. Vue.js є потужним і гнучким інструментом, що дозволяє швидко створювати інтерактивні інтерфейси користувача. Його компонентна архітектура сприяє повторному використанню коду та полегшує розробку. Крім того, використання Vue.js дозволяє підключати додаткові модулі і бібліотеки для розширення функціональності додатку.

3. **ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ**

3.1 Реалізація бази даних

Реалізація бази даних для довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків є важливим етапом проекту. База даних забезпечує збереження, організацію та доступ до інформації про фільми, сеанси, квитки, користувачів та інші сутності, необхідні для роботи системи.

Один із ключових аспектів реалізації бази даних – це визначення сутностей та взаємозв'язків між ними. Наприклад, можуть бути визначені такі сутності: фільм, сеанс, квиток, користувач, кінозал, місце, оплата тощо. Кожна з цих сутностей має свої атрибути, які визначають характеристики цієї сутності. Наприклад, у сутності "фільм" можуть бути атрибути, такі як назва фільму, тривалість, жанр, режисер тощо.

Після визначення сутностей та атрибутів, необхідно встановити взаємозв'язки між сутностями. Наприклад, між сутностями "фільм" і "сеанс" може бути встановлений зв'язок "один до багатьох", оскільки один фільм може мати багато сеансів. Також можуть бути встановлені зв'язки між сутностями "квиток" і "сеанс", "квиток" і "користувач" тощо.

Наступним кроком реалізації бази даних є визначення таблиць для кожної сутності та їх структури. Для кожного атрибуту сутності створюється відповідна колонка в таблиці. Наприклад, для сутності "фільм" може бути створена таблиця "films" з колонками "id", "назва", "тривалість", "жанр" тощо.

Після створення таблиць необхідно визначити первинні та зовнішні ключі для забезпечення цілісності даних та встановлення зв'язків між таблицями. Наприклад, в таблиці "сеанси" може бути первинний ключ "id", який буде використовуватися як зовнішній ключ у таблиці "квитки" для встановлення зв'язку між ними.

Останнім етапом реалізації бази даних є наповнення таблиць реальними даними та налаштування індексів для покращення продуктивності запитів.

Усі ці кроки реалізації бази даних вимагають використання спеціалізованих мов та інструментів для роботи з базами даних, таких як SQL (Structured Query Language) та систем управління базами даних (наприклад, MySQL, PostgreSQL, або MongoDB). Важливо також дотримуватися нормалізації бази даних для забезпечення її ефективності та гнучкості.

Реалізація бази даних для довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків грає важливу роль у забезпеченні ефективної та надійної роботи системи, збереженні та доступі до інформації.

Додатково, можуть бути застосовані техніки оптимізації запитів, такі як використання показників виконання запитів (query execution plans), кешування запитів, покращення індексації та статистики бази даних. Це дозволяє зменшити час виконання запитів та підвищити продуктивність системи в цілому.

Окрім того, важливим аспектом реалізації бази даних є забезпечення безпеки та захисту даних. Це може включати застосування різних рівнів доступу до даних, шифрування конфіденційної інформації, аудит доступу до бази даних та інші заходи для запобігання несанкціонованому доступу та втраті даних які збережені.

Важливо також розробити документацію бази даних, включаючи опис структури таблиць, зв'язків між ними, правил цілісності даних, визначення типів даних та інших важливих аспектів. Це дозволяє забезпечити зрозумілість та належне управління базою даних у майбутньому.

В цілому, реалізація бази даних для довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків є відповідальним процесом. Вона вимагає глибокого розуміння предметної області, правильного проектування структури та взаємозв'язків таблиць, а також застосування оптимізаційних технік для досягнення високої продуктивності та надійності.

Запит 1: отримати список фільмів, які відображаються в кінотеатрі в певну дату

```
SELECT m.title, s.show_date, s.show_time
FROM movies m
```

```
JOIN sessions s ON m.movie_id = s.movie_id
WHERE s.show_date = '2023-05-20';
```

Запит 2: отримати інформацію про квитки на певний фільм у певну дату та час

```
SELECT m.title, t.ticket_number, t.price, t.seat_number
FROM movies m
JOIN sessions s ON m.movie_id = s.movie_id
JOIN tickets t ON s.session_id = t.session_id
WHERE m.title = 'Avengers: Endgame' AND s.show_date = '2023-05-20' AND s.show_time = '19:30';
```

Декілька запитів до бази даних для отримання даних. Ці запити демонструють різноманітні операції, які можуть бути виконані в базі даних для довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків. Вони дозволяють отримувати інформацію про фільми, сеанси та квитки, а також здійснювати аналітику та обчислення.

3.2 Реалізація модулів системи

Для створення веб-додатку "Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків" було використано фреймворк Vue.js та ряд інших бібліотек і інструментів. Структура проекту була організована згідно зі стандартом Vue.js (рисунок 3.1). Компоненти були розподілені на окремі файли та зберігались у відповідних папках відповідно до їх функціональності. Для управління станом додатку використовувалась бібліотека Vuex.

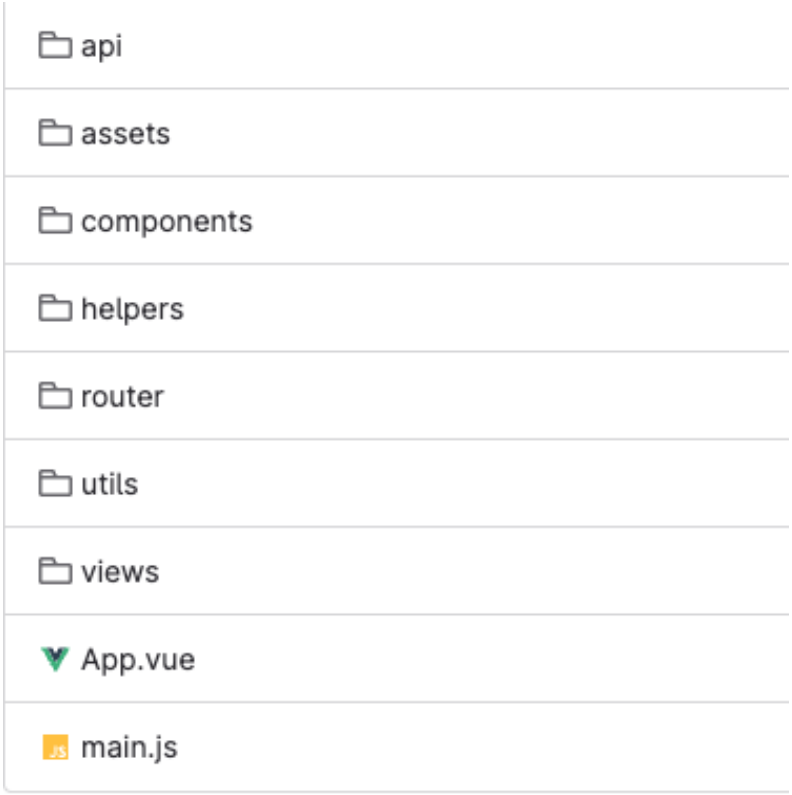


Рисунок 3.1 – Структура проекту

Для маршрутизації сторінок використовувався Vue Router. Для розмітки використовувалась бібліотека Pug. Для зв'язку з сервером та отримання даних використовувався Axios. Для реалізації онлайн-продажу квитків використовувалась платіжна система Stripe. Розробка тестів була проведена з використанням фреймворку Jest.

Один з най важливіших компонентів CatalogItem.vue. Файл "CatalogItem.vue" є компонентом Vue.js, відповідальним за відображення окремого елемента каталогу фільмів. Розпишу, що робить цей файл наступним чином. Шаблон: У файлі "CatalogItem.vue" визначений шаблон компонента, який описує структуру та розмітку елемента каталогу фільмів. Шаблон використовує HTML-подібну синтаксису Vue, де вказуються різні елементи, такі як заголовок, зображення, опис тощо. Властивості: Компонент "CatalogItem" може отримувати дані зовнішнього компонента, передаючи їх у вигляді властивостей. Наприклад, властивість "movie" використовується для

передачі даних про конкретний фільм, які будуть використовуватись для відображення інформації про фільм на елементі каталогу.

Методи: у компоненті "CatalogItem" можуть бути визначені різні методи, які виконують певні дії під час взаємодії з елементом каталогу. Наприклад, може бути метод "handleClick", який реагує на клік користувача на елементі та виконує дії, такі як перехід до сторінки фільму або виклик додаткових функцій.

Стилі: у файлі "CatalogItem.vue" також можуть бути визначені стилі для елемента каталогу. Вони можуть бути вкладені безпосередньо у компонент або використовувати зовнішні стилеві файли, які імпортуються у компонент.

У цілому, файл "CatalogItem.vue" визначає компонент, який відповідає за відображення окремого елемента каталогу фільмів. Він містить шаблон з розміткою, властивості для передачі даних та методи для взаємодії з елементом. Цей компонент може бути використаний в батьківському компоненті.

```
<template lang="pug">
  .catalog-item
    .catalog-item__wrap
      router-link.catalog-item__image(:to="{ name:'card' ,params : { id:movie.id
}}")
        img(:src="movie.image")
      .catalog-item__desc
        .catalog-item__desc-price
        .catalog-item__desc-name
      .catalog-item__detail
        .catalog-item__detail-wrap
          .catalog-item__detail-price
          .catalog-item__detail-name
            | {{movie.name}}
          .catalog-item__detail-size
          .catalog-item__detail-basket

</template>

<script>
export default {
  name: "CatalogItem",
  props: {
    movie: {
      type: Object
    }
  }
}
</script>
```

Як буде виглядати даний компонент на сторінці (рисунок 3.2).

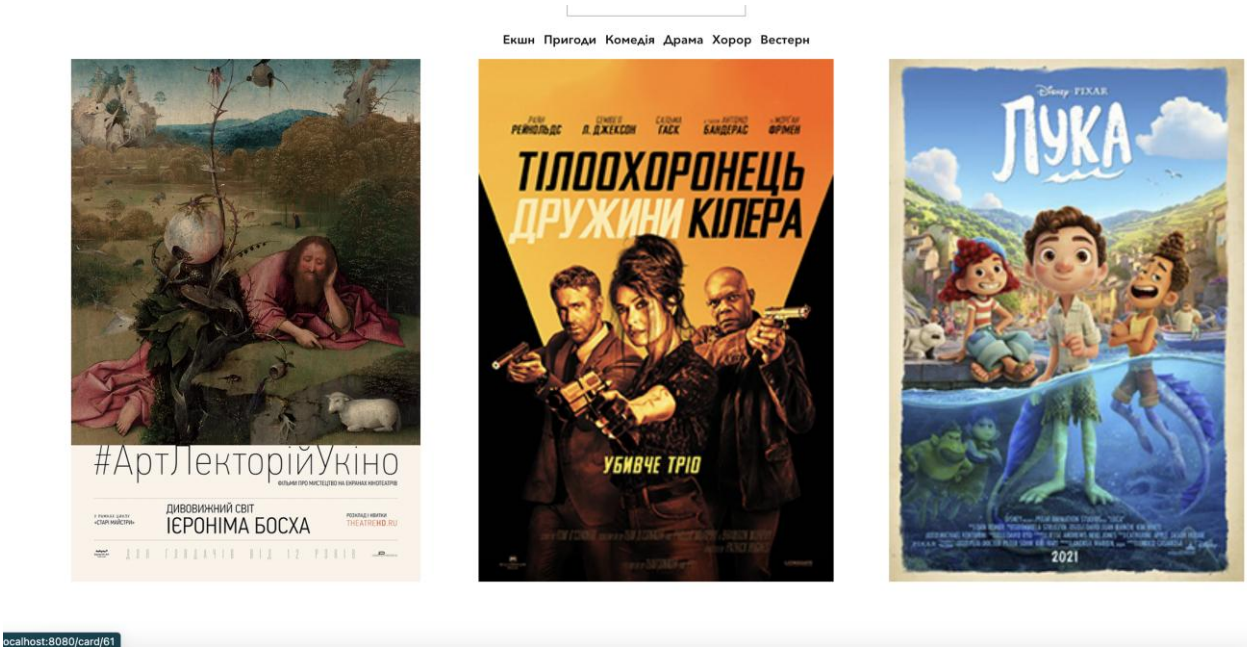


Рисунок 3.2 – Компонент картки фільму

Також нижче описані його стилі цього компонента разом з використаними міксінами для адаптації і використані статичні змінні для стилів.

```
<style lang="scss">
@import '../assets/scss/vars';
@import '../assets/scss/mixins';

.catalog-item {
  width: calc(33.33% - 30px);
  margin: 15px;
  margin-bottom: 60px;
  position: relative;
  overflow: hidden;
  transition: box-shadow .2s ease;
  will-change: box-shadow;
  z-index: 2;
  @include _1920 {
    width: calc(33.33% - 30px);
  }
  @include _1280 {
    width: calc(33.333% - 30px);
  }
  @include _768 {
    width: calc(50% - 30px);
  }
  @include _480 {
    width: 100%;
  }

  &:hover {
    box-shadow: 0 5px 25px rgba(0, 0, 0, .1);

    .catalog-item__detail {
```

```
      will-change: transform;
      transform: translateY(0);
    }
  }

  &__image {
    display: flex;
    font-size: 0;
    position: relative;
    height: 600px;
    overflow: hidden;
    width: 100%;
    background-position: center center;
    background-repeat: no-repeat;
    @include _1280 {
      height: 360px;
    }
    @include _600 {
      height: 280px;
    }
    @include _480 {
      height: 360px;
    }

    a {
      display: flex;
      position: relative;
    }

    img {
      display: flex;
      align-self: stretch;
      object-fit: contain;
      width: 100%;
    }
  }

  &__desc {
    margin-top: 15px;
    padding-bottom: 4px;

    &-price {
      margin-bottom: 16px;
      font-weight: bold;
      font-size: 22px;

      span {
        margin-right: 10px;
        font-size: 16px;
        text-decoration-line: line-through;
        color: $silver;
      }
    }

    &-name {
      font-size: 16px;

      a {
        text-decoration: none;
        color: inherit;
      }
    }
  }
```

```
  &__favorite {
    position: absolute;
    top: 16px;
    right: 16px;
    width: 34px;
    height: 30px;
    display: flex;
    justify-content: center;
    align-items: center;
    padding: 0;
    margin: 0;
    z-index: 2;

    &.selected {
      svg {
        fill: $red;
      }
    }


    svg {
      fill: #fff;
      //width: 100%;
      object-fit: cover;
    }
  }

  &__detail {
    position: absolute;
    background: #fff;
    bottom: 0;
    right: 0;
    left: 0;
    padding: 24px;
    transform: translateY(110%);
    transition: transform $trans;
    text-align: left;

    ul {
      margin-top: 15px;
    }

    &-price {
      font-weight: bold;
      font-size: 16px;
      margin-bottom: 24px;

      span {
        margin-right: 8px;
        font-size: 16px;
        text-decoration-line: line-through;
        color: $silver;
      }
    }

    &-name {
      font-size: 16px;
      margin-bottom: 8px;

      a {
        text-decoration: none;
        color: inherit;
```

```
    }
  }

  &-materials {
    font-size: 16px;
    color: $silver;
    margin-bottom: 24px;

    &_title {
      display: inline-block;
    }

    span {
      display: inline-block;
      margin-right: 5px;
    }
  }

  &-size {
    &_title {
      margin-top: 15px;
      margin-bottom: 15px;
      font-size: 16px;
    }

    li {
      min-width: 40px;
      height: 40px;
      display: inline-block;
      margin-right: 15px;
      margin-bottom: 15px;

      &:last-child {
        margin-right: 0;
      }

      input {
        display: none;

        &:checked + label {
          background: $black;
          color: #fff;
        }

        &[disabled='disabled'] {
          & ~ label {
            opacity: .5;
            cursor: default;
          }
        }
      }

      label {
        padding: 5px 10px;
        text-transform: uppercase;
        user-select: none;
        cursor: pointer;
        width: 100%;
        height: 100%;
        display: flex;
        justify-content: center;
        align-items: center;
        border: 1px solid $black;
```

```
        }
      }
    }

    &-basket {
      margin-top: 30px;
      display: flex;
      justify-content: flex-start;

      button {
        width: auto;
      }
    }
  }
}
</style>
```

Catalog.vue – це файл де відмальовується минулий компонент і проходить робота за ним. В цьому розділі визначений JavaScript-код компонента. Він починається з імпорту компонента "CatalogItem.vue" та модуля "apiCatalog". Далі слідують налаштування компонента, такі як ім'я, використовувані компоненти (у цьому випадку лише "product" - компонент "CatalogItem") та дані компонента, які містять список фільмів, значення пошуку та поточний тип сортування. Вигляд каталогу представлено на рисунок 3.3.

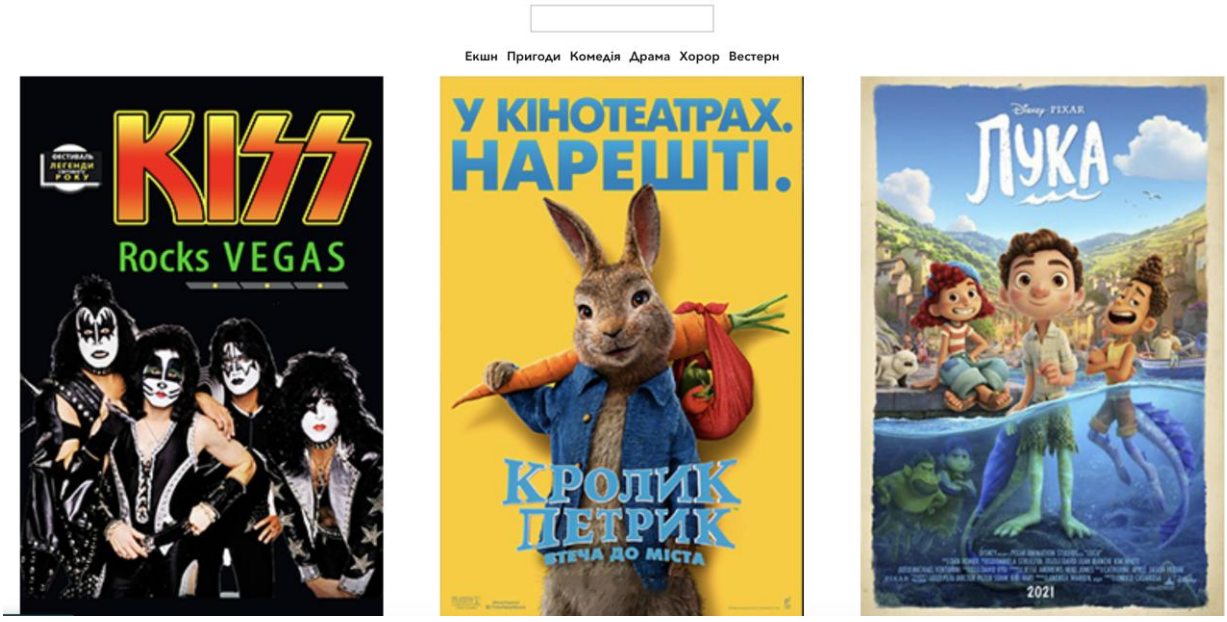


Рисунок 3.3 – Компонент каталогу

Watcher (спостерігач): В цьому розділі визначений спостерігач, який слідкує за змінами у властивості "search" і викликає метод "handleRequest" для оновлення списку фільмів залежно від нового значення пошуку.

Created hook (гачок створення): У цьому розділі використовується гачок "created", який виконується при створенні компонента. В ньому викликається метод "handleRequest" для завантаження початкового списку фільмів.

Methods (методи): В цьому розділі визначені різні методи компонента. Метод "handleRequest" виконує запит до API каталогу фільмів, передаючи тип та значення пошуку, і оновлює список фільмів. Метод "changeSort" потенційно відповідає за зміну типу сортування (але не має визначеної функціональності). Метод "sort" повертає масив об'єктів, які представляють доступні типи сортування фільмів.

Усе разом, цей код визначає компонент "CatalogGuide", який містить розмітку, дані та методи для відображення каталогу фільмів. Він взаємодіє з API каталогу, обробляє запити, оновлює дані та динамічно відображає фільми залежно від введеного пошуку та типу сортування.

Також є тут сторінка Card.vue, в якій міститься форма для вибору певного місця на певний сеанс і фільм; основний код цієї сторінки подано нижче.

```
<template lang="pug">
  .card(v-if="movies.length && moviesSessions")
    .container
      .card-wrap
        .card-left
          .card-image
            img(:src="currentMovie.image")
        .card-right
          .card-title {{currentMovie.name}}
          .card-description(v-html="currentMovie.description")
          .card-additional(v-html="currentMovie.additional")
      .card-sessions
        | Сесії:
        ul
          li(v-for="(session,index) in moviesSessions[movieId]" :key="index" )
            span {{session.showdate}}
            .card-sessions__time
              span(v-for="time in session.daytime.split(';')"
@click="orderTicket(time,session.showdate)") {{time}}
      .card-form(v-if="showBookingForm")
        .card-form__close(@click="showBookingForm = false") Close
        .card-form__wrap
```

```
          form
            label
              | Show date
              input(type="text" v-model="orderForm.showdate" readonly=true)
            label
              | Day time
              input(type="text" v-model="orderForm.daytime" readonly=true)
            .card-form__seats
              .row(v-for='(row,index) in places' :key="index")
                .seat(v-for='(seat,index) in row[1]' :key="index" ref="seat"
:class="{'free':seat.isFree,'selected':orderForm.row === row[0].row &&
orderForm.seat === seat.seat }"
@click="(e)=>{selectPlace(seat.seat,row[0].row,seat.isFree) }") {{seat.seat}}
            button(type='button' @click="sendRequest()" class="card-
form__submit" ) Request Ticket
      .card-success(v-if="showSuccessDialog")
        .card-success__close(@click="showSuccessDialog = false") Close
        .card-success__wrap
          p Ticket booking success


</template>

<script>
import cardApi from '@/api/card.js'
import {toSnakeCase} from '@/helpers'

export default {
  name: "CardView",
  data() {
    return {
      movies: [],
      moviesSessions: {},
      places: {},
      showBookingForm: false,
      showSuccessDialog: false,
      orderForm: {
        showdate: '',
        daytime: '',
        row: null,
        seat: null,
        movieId: ''
      },
    }
  },
  computed: {
    movieId() {
      return this.$route.params.id
    },
    currentMovie() {
      return this.movies[0]
    }
  },
  created() {
    this.orderForm.movieId = this.movieId
    this.init()
  },
  methods: {
    async init() {
      this.movies = await cardApi.getMovies({id: this.movieId})
      this.moviesSessions = await cardApi.getMoviesSessions({id: this.movieId})
    },
    async orderTicket(time, date) {
```

```
      this.resetForm()
      this.orderForm.daytime = time
      this.orderForm.showdate = date
      this.showBookingForm = true

      const payload = {
        id: this.movieId,
        dayTime: time,
        showdate: date
      }

      this.places = await cardApi.getBookingPlaces(payload)
    },
    selectPlace(seat, row, isFree) {
      if (this.orderForm.seat === seat && this.orderForm.row === row) {
        this.orderForm.seat = ''
        this.orderForm.row = ''
        return
      }
      if (isFree) {
        this.orderForm.seat = seat
        this.orderForm.row = row
      }
    },
    async sendRequest() {
      if (this.orderForm.seat && this.orderForm.row) {
        await cardApi.bookingTicket(toSnakeCase(this.orderForm)).then(() => {
          this.showBookingForm = false
          this.showSuccessDialog = true
        })
      }
    },
    resetForm() {
      this.orderForm.row = null
      this.orderForm.seat = null
    }
  }
}
</script>
```

В цьому розділі визначений JavaScript-код компонента. Він починається з імпорту модулів "cardApi" та "toSnakeCase" з відповідних файлів. Далі слідують налаштування компонента, такі як ім'я та дані компонента, які містять список фільмів, розклад сеансів, дані форми бронювання та стан відображення форми та діалогу успішного бронювання. Вигляд сторінки вибраного фільму представлено на рисунках 3.4 та рисунок 3.5.

Created hook (гачок створення): У цьому розділі використовується гачок "created", який виконується при створенні компонента. У ньому викликається метод "init" для отримання списку фільмів та розкладу сеансів.

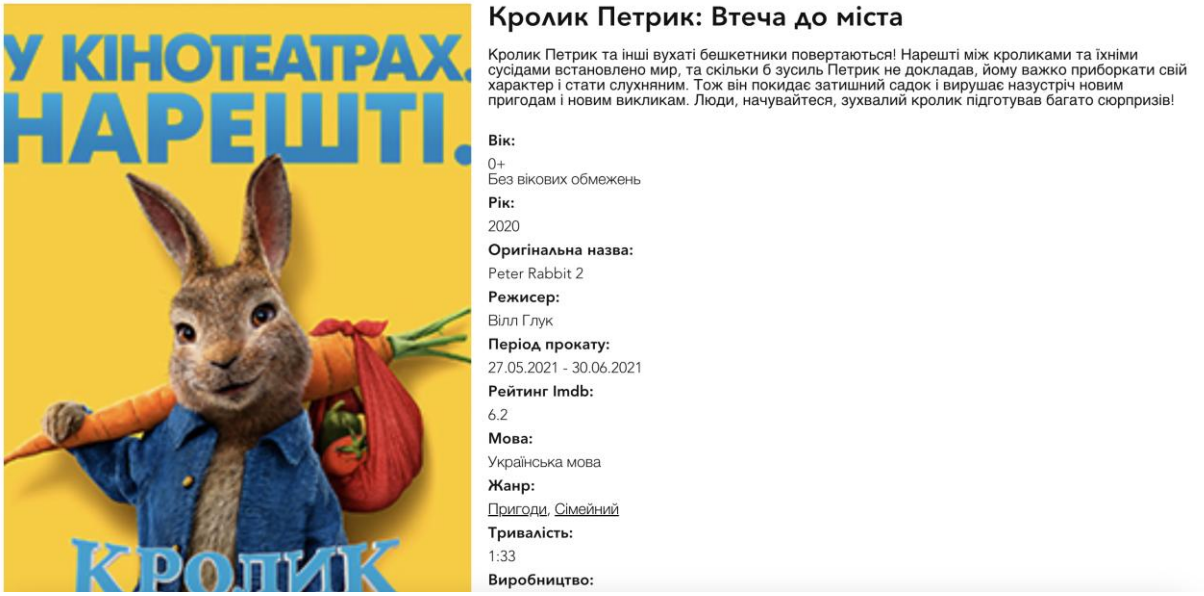


Рисунок 3.4 – Сторінка фільму

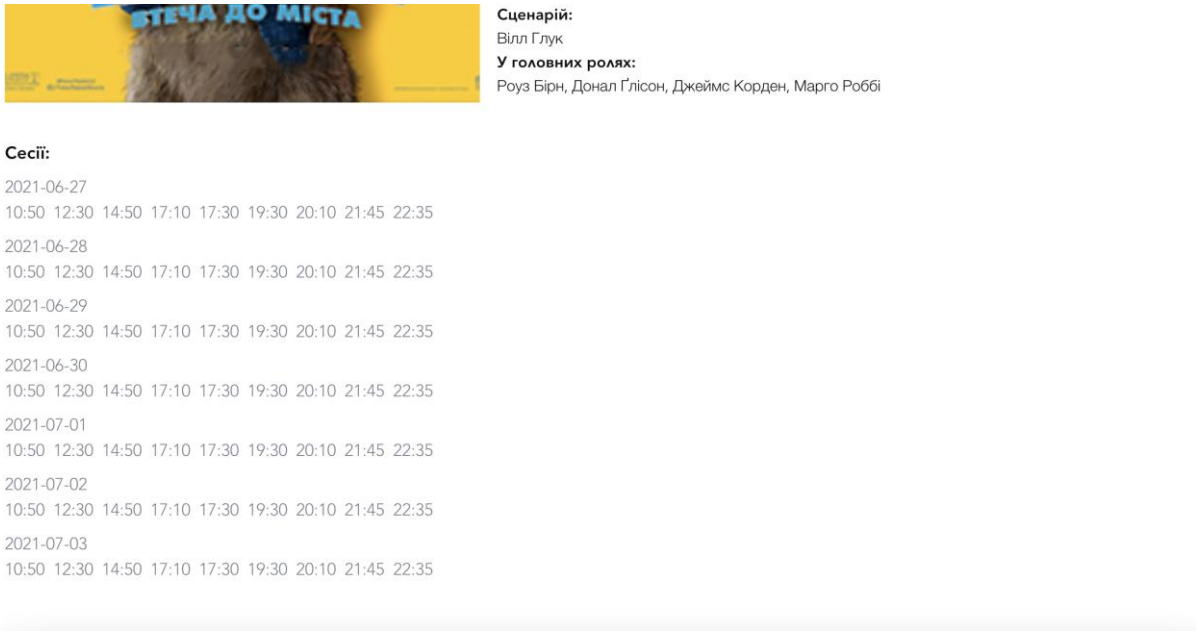


Рисунок 3.5 – Сторінка фільму (продовження)

Methods (методи): в цьому розділі визначені різні методи компонента. Метод "init" виконує запит до API для отримання списку фільмів та розкладу сеансів. Метод "orderTicket" викликається при кліку на певний сеанс та ініціює процес бронювання квитка, встановлюючи відповідні значення у формі. Метод "selectPlace" вибирає місце у залі при кліку на нього, перевіряючи його

доступність. Метод "sendRequest" відправляє запит на бронювання квитка до API, якщо вибрано місце та рядок. Метод "resetForm" скидає значення форми.

Таким чином, цей компонент відображає деталі фільму, розклад сеансів та форму для бронювання квитків. Він взаємодіє з API для отримання та відправлення даних, а також забезпечує логіку вибору місця та рядка у залі.

Імпортуються модулі cardApi та toSnakeCase з відповідних файлів.

Визначено об'єкт компонента Vue з ім'ям CardView.

Визначено властивості компонента, такі як data, computed, created і methods.

У методі init виконується запит до API для отримання списку фільмів і розкладу сеансів.

Метод orderTicket виконує необхідні дії для бронювання квитка, встановлюючи значення у формі.

Метод selectPlace обробляє вибір місця у залі і зберігає відповідні значення у формі.

Метод sendRequest відправляє запит на бронювання квитка до API, якщо відповідні поля форми заповнені.

Метод resetForm скидає значення форми бронювання.

На рисунку 3.6 можна побачити як виглядає сама форма бронювання місць, червоні місця – зайняті місця, зелене – вибране користувачем.

Цей код реалізує логіку відображення деталей фільму, розкладу сеансів та можливості бронювання квитків для вибраного фільму. Він взаємодіє з API для отримання та надсилання даних (рисунки 3.7 – 3.9).

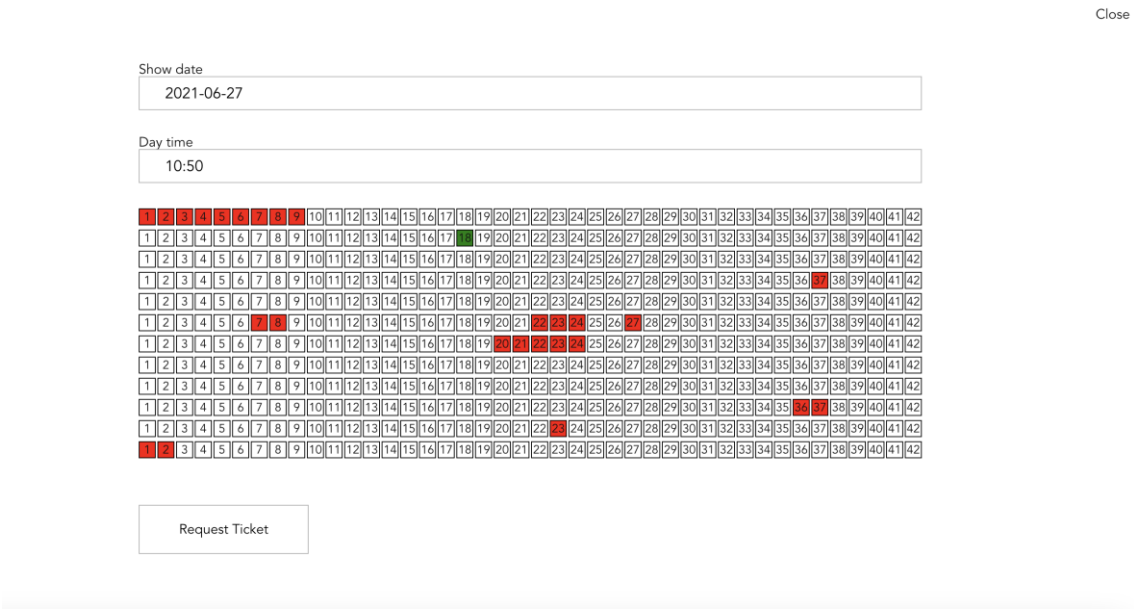


Рисунок 3.6 – Форма вибору місця

```
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  Vue.use(VueRouter)
4
5  const routes = [
6    {
7      path: '/',
8      name: 'catalog',
9      component: () => import('../views/Catalog.vue')
10   },
11   {
12     path: '/card/:id',
13     name: 'card',
14     component: () => import('../views/Card.vue')
15   },
16 ]
17
18 const router = new VueRouter({
19   mode: 'history',
20   base: process.env.BASE_URL,
21   routes
22 })
23
24 export default router
25
```

Рисунок 3.7 – Vue router

```
<template lang="pug">
  .catalog-item
    .catalog-item__wrap
      router-link.catalog-item__image(:to="{ name:'card' ,params : { id:movie.id }}")
        img(:src="movie.image")
      .catalog-item__desc
        .catalog-item__desc-price
        .catalog-item__desc-name
      .catalog-item__detail
        .catalog-item__detail-wrap
          .catalog-item__detail-price
          .catalog-item__detail-name
            | {{movie.name}}
          .catalog-item__detail-size
          .catalog-item__detail-basket

</template>
```

Рисунок 3.8 – Розмітка pug

```
import Vue from 'vue';
import axios from 'axios';
import VueAxios from 'vue-axios';

const camelcaseObjectDeep = require('camelcase-object-deep');

Vue.use(VueAxios, axios);

export const $http = axios.create({
  baseURL: process.env.VUE_APP_ROOT_API,
  headers: {
    'Accept': 'application/json',
  }
});
$http.interceptors.request.use((config) => {
  return config;
});
$http.interceptors.response.use((response) => {
    response.data = camelcaseObjectDeep(response.data);
    return response;
  }, (error) => {
    switch (error.response.status) {
      case 500: {
        break;
      }
      case 404: {
        break;
      }
      default: {
        break;
      }
    }
    return Promise.reject(camelcaseObjectDeep(error.response));
  }
);
```

Рисунок 3.9 – Конфігурація axios

Повний код (лістинг) програми подано у додатку Б.

3.3 Інструкція користувача

Користування даним додатком дуже проста, спершу користувач попадає на головну сторінку как я є одразу каталогом колекції фільмів, в навігації є вибір різних фільтрів фільмів та пошук для більш точного пошуку певного фільма. При наводжені на банер фільму, можна побачити коротки опис та його назву, при натискані на нього користувач перейде на сторінку фільма, де побачить детальний опис, що за жанр, хто актори та інше, а також під банером йому буде доступно побачити всі можливі сеанси, при натискані на один із них, відкриється форма, де потрібно вибрати місце не зайняте, червоні місця – це зайняті а зелені це вибране користувачем яке він хоче забронювати, якщо форма буде якось не так заповнена, валідація допомже, і напише що було зроблено не так в формі і користувач зможе без проблем виправити помилку в формі

3.4 Тестування довідково-інформаційної системи

Було написано тести для перевірки роботи компонентів, маршрутизації та стану додатку. В результаті роботи було створено функціональний веб-додаток, який дозволяє користувачам переглядати інформацію про кінотеатри, фільми, сеанси та квитки, а також здійснювати онлайн-продаж квитків. Використання фреймворку Vue.js та додаткових інструментів дозволило забезпечити швидку та зручну розробку додатку з оптимальною структурою коду. Тести дозволили підтвердити правильну роботу компонентів та маршрутизації додатку.

Тестування програмного забезпечення є невід'ємною складовою процесу розробки, яке спрямоване на перевірку функціональності, якості та надійності програми. Цей підпункт в проектуванні програмного забезпечення включає в

себе розробку тестових сценаріїв, виконання тестів, виявлення та виправлення помилок, а також перевірку відповідності вимогам та очікуванням користувачів.

У процесі розробки програмного забезпечення важливо проводити різноманітні види тестування, такі як модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування. Це дозволяє виявляти помилки на ранніх стадіях розробки, забезпечувати взаємодію між компонентами системи та переконатися, що програмне забезпечення працює належним чином та відповідає вимогам користувачів.

Окрім функціонального тестування, важливо також проводити навантажувальне тестування та тестування продуктивності. Це дозволяє перевірити, як програма працює під великим навантаженням, чи забезпечує вона достатню продуктивність та швидкість реакції на запити користувачів. Таке тестування допомагає виявити можливі проблеми та вузькі місця у системі, які можуть бути оптимізовані для покращення продуктивності та швидкодії.

Тестування програмного забезпечення також допомагає забезпечити високу якість та надійність. Виявлення та виправлення помилок, перевірка правильності роботи програми та відповідність вимогам допомагають покращити якість продукту та забезпечити коректну роботу системи в різних умовах. Тестування дозволяє зменшити ризик виникнення помилок та проблем у програмного забезпечення після впровадження в експлуатацію.

Таким чином, тестування програмного забезпечення є важливою складовою розробки, яка допомагає перевірити та підтвердити якість, функціональність та продуктивність програми. Воно дозволяє виявити та усунути помилки, забезпечити надійність та швидкодію системи, а також забезпечити відповідність програмного забезпечення вимогам та очікуванням.

Тестування програмного забезпечення має кілька важливих цілей. По-перше, воно допомагає впевнитися, що програма виконує свої функції згідно з очікуваннями. Це означає, що всі функції програми працюють належним чином, виконують необхідні дії та повертають правильні результати. Тестування

допомагає виявити та виправити помилки, що можуть з'явитися під час виконання програми.

По-друге, тестування допомагає перевірити продуктивність програмного забезпечення. Це означає, що програма працює ефективно та швидко навіть при великому обсязі даних або під час одночасної роботи багатьох користувачів. Тестування продуктивності дозволяє виявити можливі проблеми з продуктивністю та забезпечити оптимальну роботу програми в різних умовах.

Крім того, тестування допомагає забезпечити масштабованість програмного забезпечення. Це означає, що програма може ефективно працювати при збільшенні обсягу даних, кількості користувачів або розміру системи. Тестування масштабованості дозволяє виявити можливі обмеження та проблеми, пов'язані з масштабуванням програми, і прийняти необхідні заходи для їх вирішення.

Загалом, тестування програмного забезпечення є важливим етапом в розробці, який допомагає забезпечити якість, продуктивність та масштабованість програми. Це процес, що включає розробку тестових сценаріїв, виконання тестів, виявлення та виправлення помилок, а також перевірку відповідності програми вимогам та очікуванням користувачів. Правильно виконане тестування дозволяє покращити якість та надійність програмного забезпечення, що, в свою чергу, сприяє успіху проекту і задоволенню користувачів які будуть користуватись цим додатком.

Тестування програмного забезпечення включає в себе різні види тестів, такі як модульні тести, інтеграційні тести, системні тести та інші. Кожен з цих видів тестів спрямований на перевірку певних аспектів програми, від окремих функцій до повного функціоналу системи. Це дозволяє виявити помилки, недоліки та несправності, що можуть впливати на роботу програми.

Окрім виявлення помилок, тестування також допомагає підтвердити, що програма відповідає вимогам та специфікаціям, визначеним на етапі проектування. Це забезпечує відповідність функціональності програми очікуванням користувачів та бізнес-потребам. Тестування також сприяє

забезпеченню стабільності та надійності програми, допомагаючи виявити та усунути можливі проблеми з витоком пам'яті, недостатньою продуктивністю або некоректною обробкою вхідних даних.

У процесі тестування використовуються різні методи та інструменти, включаючи автоматизовані тестувальні фреймворки, які допомагають спростити та прискорити процес тестування. Це забезпечує ефективне виконання тестів і швидке виявлення проблем.

Загалом, тестування програмного забезпечення є необхідним етапом в розробці проекту, який допомагає забезпечити якість, надійність та відповідність програмного продукту вимогам користувачів. Використання різних видів тестів та відповідних інструментів дозволяє ефективно виявляти та виправляти помилки, забезпечуючи оптимальну роботу програми та задоволення користувачів.

3.5 Висновок

Під час програмної реалізації та тестування довідково-інформаційної системи кінотеатру з реалізацією онлайн-продажу квитків було здійснено значний обсяг роботи, що включав розробку функціональності, створення бази даних, налаштування серверного середовища та інтеграцію з зовнішніми сервісами для отримання даних.

Процес програмної реалізації включав в себе розробку фронтенду та бекенду системи. На фронтенді було використано Vue.js для створення інтерактивного інтерфейсу користувача, який забезпечував зручний та привабливий вигляд додатку. Бекенд реалізовувався з використанням Node.js та Express.js, що дозволило створити надійний серверний шар системи.

Протягом реалізації системи проводилися регулярні тестування для перевірки функціональності, виявлення та виправлення помилок. Було

використано автоматизоване тестування, таке як модульні тести та інтеграційні тести, а також ручне тестування для перевірки коректності роботи системи з точки зору користувача.

В результаті програмної реалізації та тестування була створена функціональна та надійна довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків. Система має зручний та інтуїтивний інтерфейс користувача, що дозволяє швидко та зручно здійснювати пошук фільмів, переглядати розклад сеансів та придбавати квитки онлайн. Тестування системи підтвердило її стабільну та надійну роботу, а також виявило та виправило помилки, що дозволило забезпечити якісну роботу системи з максимальною ефективністю.

У цілому, програмна реалізація та тестування системи пройшли успішно, що дозволило отримати готовий продукт, який задовольняє вимоги та потреби кінотеатру та його клієнтів.

ВИСНОВКИ

Загальний висновок цієї кваліфікаційної роботи полягає в тому, що було успішно розроблено довідково-інформаційну систему кінотеатру з реалізацією онлайн-продажу квитків з використанням Vue.js, Vue Router та Vuex. Було проведено аналіз предметної області та вибір необхідних інструментів для розробки. Була створена структура проекту, що включає в себе різні модулі, які відповідають за різні функції проекту. Були реалізовані основні функції проекту, такі як перегляд фільмів, пошук, реєстрація та вхід користувачів, купівля та оплата квитків, відгуки та рейтинги. Крім того, було розроблено тестові сценарії для перевірки функціональності проекту. В результаті роботи було успішно досягнуто всіх поставлених цілей та завдань. Розроблене програмне забезпечення є готовим до використання та може бути вдосконалене або розширене у майбутньому. Основна перевага проекту полягає в його простоті та інтуїтивному інтерфейсі, що робить його зручним для використання для будь-якого користувача. У процесі розробки дипломної роботи я отримав багато корисного досвіду в роботі з Vue.js, Vuex та Vue Router, що дозволить мені в подальшому ефективно використовувати ці інструменти для розробки інших проектів.

Також, розробка тестів дала мені більш глибоке розуміння функціональності проекту та його потенційних проблем, що дозволить мені забезпечувати якість моїх майбутніх проектів. Структура проекту була організована за принципом розбиття на компоненти та модулі, що забезпечило легкість розробки, тестування та розширення. Також було забезпечено безпеку даних за допомогою збереження паролів у хешованому вигляді. У результаті роботи було реалізовано функціонал для замовлення квитків онлайн, перегляду розкладу сеансів, перегляду опису фільмів та їхньої тривалості, пошук за жанрами, а також додано можливість редагування та видалення фільмів з бази даних. Для поліпшення роботи додатку можна додати функціонал для

авторизації користувачів, використання більш ефективних алгоритмів пошуку та сортування фільмів, а також розширення бази даних з інформацією про фільми та кінотеатри. У цілому, проект був успішно реалізований, і здійснюватиме зручну та ефективну роботу для користувачів кінотеатру, які бажають замовити квитки та отримати інформацію про фільми

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Internet movie database. URL: https://www.imdb.com/ (дата звернення 11.02.2023)
2. Rotten Tomatoes. URL: https://www.rottentomatoes.com/ (дата звернення 11.02.2023)
3. Wikipedia Client–server model. URL: https://en.wikipedia.org/wiki/Client%E2%80%93server_model (дата звернення 14.02.2023)
4. Vue.js офіційна документація. URL: https://vuejs.org/ (дата звернення: 19.01.2023).
5. Vue Router офіційна документація. URL: https://router.vuejs.org/ (дата звернення: 19.01.2023).
6. Vue CLI офіційна документація. URL: https://cli.vuejs.org/ (дата звернення: 19.01.2023).
7. UML Distilled: A Brief Guide to the Standard Object Modeling Language - URL: https://www.amazon.com/UML-Distilled-Standard-Modeling-Language/dp/0321193687 (дата звернення: 19.01.2023).
8. UML 2.0 in a Nutshell URL: https://www.amazon.com/UML-2-0-Nutshell-OReilly/dp/0596007957 (дата звернення: 19.01.2023).
9. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development URL: https://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0131489062 (дата звернення: 19.01.2023).
10. UML Classroom: An Introduction to Object-Oriented Modeling URL: https://www.springer.com/gp/book/9783319127415 (дата звернення: 19.01.2023).

11. Назаренко Л. Адаптивна організаційна структура управління як чинник ефективного менеджменту організації. New pedagogical thought. 2022. Т. 109, № 1. С. 8–14. URL: https://doi.org/10.37026/2520-6427-2021-109-1-8-14 (дата звернення: 01.03.2023).
12. Плєхова Г., Суханова Н., Левтеров А. Кібербезпека: загрози, рішення. Theoretical foundations in economics and management. 2022. С. 681–692. URL: https://doi.org/10.46299/isg.2022.mono.econ.2.9.6 (дата звернення: 02.03.2023).
13. UML for the IT Business Analyst URL: https://www.amazon.com/UML-IT-Business-Analyst-Techniques/dp/1598638688 (дата звернення: 19.01.2023).
14. The Unified Modeling Language User Guide URL: https://www.amazon.com/Unified-Modeling-Language-User-Guide/dp/0201571684 (дата звернення: 19.01.2023).
15. UML and the Unified Process: Practical Object-Oriented Analysis and Design - URL: https://www.amazon.com/UML-Unified-Process-Practical-Object-Oriented/dp/0201730448 (дата звернення: 19.01.2023).
16. UML for Database Design URL: https://www.amazon.com/UML-Database-Design-Eric-Naiburg/dp/1558606326 (дата звернення: 19.01.2023).
17. UML Weekend Crash Course URL: https://www.amazon.com/UML-Weekend-Crash-Course/dp/076454951X (дата звернення: 19.01.2023).
18. PHP for the Web QuickStart Guide URL: https://www.peachpit.com/store/php-for-the-web-visual-quickstart-guide-9780134291253 (дата звернення: 19.01.2023).
19. Modern PHP: New Features and Good Practices URL: https://www.oreilly.com/library/view/modern-php/9781491904992 (дата звернення: 19.01.2023).
20. PHP Objects, Patterns, and Practice URL: https://www.apress.com/gp/book/9781430260318 (дата звернення: 19.01.2023).

21. PHP and MySQL Web Development URL: https://www.informit.com/store/php-and-mysql-web-development-9780321833891 (дата звернення: 19.01.2023).
22. PHP Advanced and Object-Oriented Programming URL: https://www.peachpit.com/store/php-advanced-and-object-oriented-programming-9780321832184 (дата звернення: 3.01.2023).
23. Database Systems: Design, Implementation, and Management URL: https://www.amazon.com/Database-Systems-Implementation-Management-Coronel/dp/1305627482 (дата звернення: 5.01.2023).
24. Database Design and Implementation: A Practical Introduction Using Oracle SQL URL: https://www.springer.com/gp/book/9783030693496 (дата звернення: 19.01.2023).
25. SQL and Relational Theory: How to Write Accurate SQL Code URL: https://www.amazon.com/SQL-Relational-Theory-Write-Accurate/dp/0596523068 (дата звернення: 19.01.2023).
26. Database System Concepts URL: https://www.amazon.com/Database-System-Concepts-Abraham-Silberschatz/dp/0073523321 (дата звернення: 19.01.2023).
27. JavaScript: The Good Parts URL: https://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742 (дата звернення: 19.01.2023).
28. Eloquent JavaScript: A Modern Introduction to Programming URL: https://eloquentjavascript.net/ (дата звернення: 19.01.2023).
29. CSS Secrets: Better Solutions to Everyday Web Design Problems URL: https://www.amazon.com/CSS-Secrets-Lea-Verou/dp/1449372635 (дата звернення: 19.01.2023).
30. HTML and CSS: Design and Build Websites URL: https://www.amazon.com/HTML-CSS-Design-Build-Websites/dp/1118008189 (дата звернення: 19.01.2023).

31. React: Up and Running: Building Web Applications URL: https://www.amazon.com/React-Up-Running-Building-Applications/dp/1491931825 (дата звернення: 21.01.2023).
32. Vue.js: Up and Running: Building Accessible and Performant Web Apps URL: https://www.amazon.com/Vue-js-Running-Building-Accessible-Performant/dp/1491997249 (дата звернення: 24.01.2023).
33. JavaScript: The Definitive Guide URL: https://www.amazon.com/JavaScript-Definitive-Guide-Activate-Guides/dp/0596805527 (дата звернення: 24.01.2023).
34. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics URL: https://www.amazon.com/Learning-Web-Design-Beginners-JavaScript/dp/1491960205 (дата звернення: 24.01.2023).
35. JavaScript and JQuery: Interactive Front-End Web Development URL: https://www.amazon.com/JavaScript-JQuery-Interactive-Front-End-Development/dp/1118531647 (дата звернення: 03.03.2023).
36. CSS: The Missing Manual URL: https://www.amazon.com/CSS-Missing-Manual-David-McFarland/dp/1491918055 (дата звернення: 19.01.2023).
37. Angular Development with TypeScript URL: https://www.amazon.com/Angular-Development-TypeScript-Yakov-Fain/dp/1617295349 (дата звернення: 03.03.2023).
38. CSS: The Definitive Guide URL: https://www.amazon.com/CSS-Definitive-Guide-Eric-Meyer/dp/1449393195 (дата звернення: 03.03.2023).
39. You Don't Know JS URL: https://github.com/getify/You-Dont-Know-JS (дата звернення: 03.03.2023).
40. Design Systems: A Practical Guide to Creating Design Languages for Digital Products URL: https://www.amazon.com/Design-Systems-Practical-Creating-Language/dp/1491921560 (дата звернення: 14.04.2023).
41. The Pragmatic Programmer: Your Journey to Mastery URL: https://www.amazon.com/Pragmatic-Programmer-Journey-Mastery/dp/0135957052 (дата звернення: 14.04.2023).

42. Web Design with HTML, CSS, JavaScript and jQuery Set URL: https://www.amazon.com/Web-Design-HTML-JavaScript-jQuery/dp/1118907442 (дата звернення: 14.04.2023).
43. Vue.js 3 Cooking URL: https://www.amazon.com/Vue-js-3-Cookbook-Heitor-Ribeiro/dp/1800203191 (дата звернення: 14.04.2023).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: «Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків ».

2 Призначення розробки

2.1 Функціональне призначення

Функціональне призначення цієї роботи полягає в розробці довідково-інформаційної системи для кінотеатру з реалізацією онлайн-продажу квитків. Основною метою проекту є надання користувачам зручного та доступного інструменту для отримання інформації про фільми, сеанси та придбання квитків через Інтернет.

2.2 Експлуатаційне призначення

Ця система призначена для використання клієнтами кінотеатру, які можуть швидко та зручно отримувати інформацію про фільми, розклад сеансів та придбавати квитки з будь-якого пристрою з доступом до Інтернету.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати можливості виконання таких функцій:

Пошук сеансів;

Вибір місць на певний сеанс;

Фільтрація та сортування прокатних фільмів;

Функція відображення зайнятих місць

Можливість користування як на планшетах так і на телефонах або з моніторами в розширені 4к

3.2 Вимоги до надійності

Додаток повинен забезпечувати наступні вимоги до надійності:

Висока доступність: Забезпечення неперервної роботи системи та доступності сервісу для користувачів. Резервування серверів, балансування навантаження та використання механізмів реплікації даних для запобігання відмовам у роботі та забезпечення швидкого відновлення системи в разі виникнення проблем.

Безпека даних: Забезпечення захисту особистої інформації користувачів, включаючи дані про платежі, контактну інформацію та інші особисті дані. Використання шифрування для передачі та зберігання даних, реалізація механізмів автентифікації та авторизації користувачів.можливість самовідновлення після збоїв;

Тестування та перевірка: Систематична перевірка та тестування різних аспектів системи, включаючи функціональність, стабільність, безпеку та продуктивність. Використання автоматизованих тестів, що дозволяють виявляти помилки та дефекти в ранніх етапах розробки.

Резервне копіювання та відновлення: Регулярне створення резервних копій даних системи та забезпечення можливості швидкого відновлення в разі втрати або пошкодження даних

3.3 Умови експлуатації та вимоги до технічних засобів

Доступ до Інтернету: Для використання системи потрібний доступ до Інтернету, оскільки вона працює в онлайн-режимі. Користувачі зможуть отримувати інформацію про фільми та придбати квитки з будь-якого пристрою, що має доступ до Інтернету, такого як комп'ютер, планшет або смартфон.

Сумісність з браузерами: Розроблена система повинна бути сумісною з різними веб-браузерами, що використовуються користувачами. Незалежно від того, чи вони використовують Google Chrome, Mozilla Firefox, Safari або Інтернет-Explorer, система має працювати із застосуванням будь-якого з цих браузерів.

Мінімальні вимоги до обладнання: Для користування системою необхідно мати пристрій, який відповідає мінімальним вимогам до обладнання. Це може

бути сучасний комп'ютер, планшет або смартфон, які мають достатній обсяг оперативної пам'яті, швидкий процесор та достатній обсяг внутрішньої пам'яті.

Захист від несанкціонованого доступу: Система повинна мати вбудовані механізми безпеки для запобігання несанкціонованому доступу до конфіденційної інформації. Це може включати механізми автентифікації, шифрування даних, захист від атак на перехоплення та інші заходи безпеки.

Для успішного запуску додатку пристрій повинен відповідати наступним рекомендованим вимогам:

Windows 10;

будь-який чотирьохядерний процесор;

1 ГБ ОЗП;

500 МБ постійної пам’яті.

3.4 Вимоги до інформаційної та програмної сумісності

При розробці додатку використовуватиметься високорівнева об’єктно-орієнтована мова C#, а також Xamarin – фреймворк з відкритим кодом, призначений для створення додатків для iOS, Android та Windows на платформі .NET. В якості СКБД використовуватиметься SQLite, а Entity Framework Core буде використовуватись як об’єктно-орієнтована технологія для спрощення роботи із нею.

3.5 Спеціальні вимоги

Програма повинна мати зручний та приємний матеріальний дизайн інтерфейсу, зрозумілий для будь-якого користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

– текст програми;

– опис програми;

– технічне завдання;

– керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки довідково-інформаційної системи визначенні календарним графіком дипломного проектування.

ДОДАТОК Б

(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Папка api card.js

```
import {$http} from '@/utils/http'

export default {
  async getMovies({id = ''}) {
    const {data: {data}} = await $http.get(`/movies?movie_id=${id}`);
    return data
  },
  async getMoviesSessions({id = ''}) {
    const {data: {data}} = await $http.get(`/movieShows?movie_id=${id}`);
    return data
  },
  async getBookingPlaces({id = '', dayTime = '', showdate = ''}) {
    const {data: {data}} = await
$http.get(`/showPlaces?movie_id=${id}&daytime=${dayTime}&showdate=${showdate}`);
    return data
  },
  async bookingTicket(payload) {
    const {data: {data}} = await $http.post(`/bookPlace`, payload);
    return data
  },
}
```

Папка api catalog.js

```
import {$http} from '@/utils/http'

export default {
  async getList(genres,name) {
    const {data: {data}} = await
$http.get(`/movies?genres=${genres}&name=${name}`);
    return data
  },
}
```

Папка assets -> scss -> _mixins.scss

```
@mixin _4000 {
  @media (min-width: 4000px){
    @content;
  }
}
@mixin _3000 {
  @media (min-width: 3000px){
    @content;
```

```
  }
}
@mixin _2000 {
  @media (min-width: 2000px){
    @content;
  }
}
@mixin _1920 {
  @media (min-width: 1920px){
    @content;
  }
}
@mixin _1280 {
  @media (max-width: 1280px){
    @content;
  }
}
@mixin _1170 {
  @media (max-width: 1170px){
    @content;
  }
}
@mixin _979 {
  @media (max-width: 979px){
    @content;
  }
}
@mixin _768 {
  @media (max-width: 768px){
    @content;
  }
}
@mixin _600 {
  @media (max-width: 600px){
    @content;
  }
}
@mixin _480 {
  @media (max-width: 480px){
    @content;
  }
}
@mixin _350{
  @media (max-width: 350px){
    @content;
  }
}
```

Папка assets -> scss -> _reset.scss

```
html, body, div, span, applet, object, iframe,
blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, img, ins, kbd, q, s, samp,
small, strike, sub, sup, tt, var, u, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
```

```
menu, nav, output, ruby, section, summary,
time, mark, audio, video, input, button {
  margin: 0;
  padding: 0;
  border: 0;
  font: inherit;
  font-size: 100%;
  vertical-align: baseline;
}
h1, h2, h3, h4, h5, h6, p, b, strong, i, a, span{
  margin: 0;
  padding: 0;
  border: 0;
  line-height: 1;
}
html {
  line-height: 1;
}
ol, ul {
  list-style: none;
}

table {
  border-collapse: collapse;
  border-spacing: 0;
}

caption, th, td {
  text-align: left;
  font-weight: normal;
  vertical-align: middle;
}

q, blockquote {
  quotes: none;
}
q:before, q:after, blockquote:before, blockquote:after {
  content: "";
  content: none;
}
*, :after, :before {
  box-sizing: content-box!important;
}

a img {
  border: none;
}
img{
  max-width: 100%;
}
button{
  border: none;
  background: transparent;
  cursor: pointer;
}
article, aside, details, figcaption, figure, footer, header, hgroup, main, menu,
nav, section, summary {
  display: block;
}
*{
  box-sizing: border-box!important;
  outline: none!important;
  -webkit-appearance: none!important;
  appearance: none!important;
```

```
  -webkit-tap-highlight-color: transparent;
}
a[href^=tel]{
  color: #000;
}
.clearfix::after{
  content: " ";
  clear: both;
  display: block;
  height: 0;
  overflow: hidden;
  visibility: hidden;
}
*::-webkit-input-placeholder {
  color: #b3b3b3;
  opacity: 1;
}
*:-moz-placeholder {
  color: #b3b3b3;
  opacity: 1;
}
*::-moz-placeholder {
  color: #b3b3b3;
  opacity: 1;
}

*:-ms-input-placeholder {
  color: #b3b3b3;
  opacity: 1;
}
input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  /* display: none; <- Crashes Chrome on hover */
  -webkit-appearance: none;
  margin: 0; /* <-- Apparently some margin are still there even though it's hidden
*/
}
input[type="search"]::-webkit-search-decoration,
input[type="search"]::-webkit-search-cancel-button,
input[type="search"]::-webkit-search-results-button,
input[type="search"]::-webkit-search-results-decoration {
  -webkit-appearance:none;
}
input{}
```

Папка assets -> scss -> main.scss

```
@import "vars";
@import "mixins";
@import "reset";

.container {
  width: 100%;
  max-width: 1470px;
  margin: 0 auto;
  padding-left: 30px;
  padding-right: 30px;
  position: relative;
  @include _1920 {
    max-width: 1800px;
```

```
  }
  @include _2000 {
    max-width: 1920px;
  }
  @include _3000 {
    max-width: 3000px;
  }
}

.wrapper {
  width: 100%;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  color: $black;
  justify-content: space-between;
  padding-top: 50px;
  padding-bottom: 50px;

  &-top {
    width: 100%;
  }

  &-bottom {
    width: 100%;
  }
}

#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}
```

Папка components CatalogItem.vue

```
<template lang="pug">
  .catalog-item
    .catalog-item__wrap
      router-link.catalog-item__image(:to="{ name:'card' ,params : { id:movie.id
}}")
        img(:src="movie.image")
      .catalog-item__desc
        .catalog-item__desc-price
        .catalog-item__desc-name
      .catalog-item__detail
        .catalog-item__detail-wrap
          .catalog-item__detail-price
          .catalog-item__detail-name
            | {{movie.name}}
          .catalog-item__detail-size
          .catalog-item__detail-basket

</template>

<script>
export default {
  name: "CatalogItem",
```

```
  props: {
    movie: {
      type: Object
    }
  }
}
</script>

<style lang="scss">
@import '../assets/scss/vars';
@import '../assets/scss/mixins';

.catalog-item {
  width: calc(33.33% - 30px);
  margin: 15px;
  margin-bottom: 60px;
  position: relative;
  overflow: hidden;
  transition: box-shadow .2s ease;
  will-change: box-shadow;
  z-index: 2;
  @include _1920 {
    width: calc(33.33% - 30px);
  }
  @include _1280 {
    width: calc(33.333% - 30px);
  }
  @include _768 {
    width: calc(50% - 30px);
  }
  @include _480 {
    width: 100%;
  }

  &:hover {
    box-shadow: 0 5px 25px rgba(0, 0, 0, .1);

    .catalog-item__detail {
      will-change: transform;
      transform: translateY(0);
    }
  }

  &__image {
    display: flex;
    font-size: 0;
    position: relative;
    height: 600px;
    overflow: hidden;
    width: 100%;
    background-position: center center;
    background-repeat: no-repeat;
    @include _1280 {
      height: 360px;
    }
    @include _600 {
      height: 280px;
    }
    @include _480 {
      height: 360px;
    }

    a {
      display: flex;
```

```
    position: relative;
  }

  img {
    display: flex;
    align-self: stretch;
    object-fit: contain;
    width: 100%;
  }
}

&__desc {
  margin-top: 15px;
  padding-bottom: 4px;

  &-price {
    margin-bottom: 16px;
    font-weight: bold;
    font-size: 22px;

    span {
      margin-right: 10px;
      font-size: 16px;
      text-decoration-line: line-through;
      color: $silver;
    }
  }

  &-name {
    font-size: 16px;

    a {
      text-decoration: none;
      color: inherit;
    }
  }
}

&__favorite {
  position: absolute;
  top: 16px;
  right: 16px;
  width: 34px;
  height: 30px;
  display: flex;
  justify-content: center;
  align-items: center;
  padding: 0;
  margin: 0;
  z-index: 2;

  &.selected {
    svg {
      fill: $red;
    }
  }


  svg {
    fill: #fff;
    //width: 100%;
    object-fit: cover;
  }
}
```

```
&__detail {
  position: absolute;
  background: #fff;
  bottom: 0;
  right: 0;
  left: 0;
  padding: 24px;
  transform: translateY(110%);
  transition: transform $trans;
  text-align: left;

  ul {
    margin-top: 15px;
  }

  &-price {
    font-weight: bold;
    font-size: 16px;
    margin-bottom: 24px;

    span {
      margin-right: 8px;
      font-size: 16px;
      text-decoration-line: line-through;
      color: $silver;
    }
  }

  &-name {
    font-size: 16px;
    margin-bottom: 8px;

    a {
      text-decoration: none;
      color: inherit;
    }
  }

  &-materials {
    font-size: 16px;
    color: $silver;
    margin-bottom: 24px;

    &_title {
      display: inline-block;
    }

    span {
      display: inline-block;
      margin-right: 5px;
    }
  }

  &-size {
    &_title {
      margin-top: 15px;
      margin-bottom: 15px;
      font-size: 16px;
    }

    li {
      min-width: 40px;
      height: 40px;
```

```
        display: inline-block;
        margin-right: 15px;
        margin-bottom: 15px;

        &:last-child {
          margin-right: 0;
        }

        input {
          display: none;

          &:checked + label {
            background: $black;
            color: #fff;
          }

          &[disabled='disabled'] {
            & ~ label {
              opacity: .5;
              cursor: default;
            }
          }
        }

        label {
          padding: 5px 10px;
          text-transform: uppercase;
          user-select: none;
          cursor: pointer;
          width: 100%;
          height: 100%;
          display: flex;
          justify-content: center;
          align-items: center;
          border: 1px solid $black;
        }
      }
    }

    &-basket {
      margin-top: 30px;
      display: flex;
      justify-content: flex-start;

      button {
        width: auto;
      }
    }
  }
}
</style>
```

Папка helpers index.js

```
import { snakeCase } from 'lodash'
const snakeKeysObjectDeep = require('snakecase-keys')

export function toSnakeCase (data, type='object') {
  const payload = type === 'object' ? {} : []
  Object.entries(data).forEach(e => {
```

```
    const [key, value] = e
    if(type === 'object'){
      payload[snakeCase(key)] = value
    }else{
      payload.push(snakeKeysObjectDeep(value))
    }

  })
  return payload
}
```

Папа route index.js

```
import Vue from 'vue'
import VueRouter from 'vue-router'
Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'catalog',
    component: () => import('../views/Catalog.vue')
  },
  {
    path: '/card/:id',
    name: 'card',
    component: () => import('../views/Card.vue')
  },
]

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

export default router
```

Папка utils http.js

```
import Vue from 'vue';
import axios from 'axios';
import VueAxios from 'vue-axios';

const camelcaseObjectDeep = require('camelcase-object-deep');

Vue.use(VueAxios, axios);

export const $http = axios.create({
  baseURL: process.env.VUE_APP_ROOT_API,
  headers: {
    'Accept': 'application/json',
  }
});
$http.interceptors.request.use((config) => {
  return config;
});
```

```
$http.interceptors.response.use((response) => {
    response.data = camelcaseObjectDeep(response.data);
    return response;
  }, (error) => {
    switch (error.response.status) {
      case 500: {
        break;
      }
      case 404: {
        break;
      }
      default: {
        break;
      }
    }
    return Promise.reject(camelcaseObjectDeep(error.response));
  }
);
```

Папка views Card.vue

```
<template lang="pug">
  .card(v-if="movies.length && moviesSessions")
    .container
      .card-wrap
        .card-left
          .card-image
            img(:src="currentMovie.image")
        .card-right
          .card-title {{currentMovie.name}}
          .card-description(v-html="currentMovie.description")
          .card-additional(v-html="currentMovie.additional")
      .card-sessions
        | Сеcіі:
        ul
          li(v-for="(session,index) in moviesSessions[movieId]" :key="index" )
            span {{session.showdate}}
            .card-sessions__time
              span(v-for="time in session.daytime.split(';')"
@click="orderTicket(time,session.showdate)") {{time}}
      .card-form(v-if="showBookingForm")
        .card-form__close(@click="showBookingForm = false") Close
        .card-form__wrap
          form
            label
              | Show date
              input(type="text" v-model="orderForm.showdate" readonly=true)
            label
              | Day time
              input(type="text" v-model="orderForm.daytime" readonly=true)
            .card-form__seats
              .row(v-for='(row,index) in places' :key="index")
                .seat(v-for='(seat,index) in row[1]' :key="index" ref="seat"
:class="{'free':seat.isFree,'selected':orderForm.row === row[0].row &&
orderForm.seat === seat.seat }"
@click="(e)=>{selectPlace(seat.seat,row[0].row,seat.isFree) }") {{seat.seat}}
            button(type='button' @click="sendRequest()" class="card-form__submit"
) Request Ticket
      .card-success(v-if="showSuccessDialog")
        .card-success__close(@click="showSuccessDialog = false") Close
```

```
        .card-success__wrap
          p Ticket booking success


</template>

<script>
import cardApi from '@/api/card.js'
import {toSnakeCase} from '@/helpers'

export default {
  name: "CardView",
  data() {
    return {
      movies: [],
      moviesSessions: {},
      places: {},
      showBookingForm: false,
      showSuccessDialog: false,
      orderForm: {
        showdate: '',
        daytime: '',
        row: null,
        seat: null,
        movieId: ''
      },
    }
  },
  computed: {
    movieId() {
      return this.$route.params.id
    },
    currentMovie() {
      return this.movies[0]
    }
  },
  created() {
    this.orderForm.movieId = this.movieId
    this.init()
  },
  methods: {
    async init() {
      this.movies = await cardApi.getMovies({id: this.movieId})
      this.moviesSessions = await cardApi.getMoviesSessions({id: this.movieId})
    },
    async orderTicket(time, date) {
      this.resetForm()
      this.orderForm.daytime = time
      this.orderForm.showdate = date
      this.showBookingForm = true

      const payload = {
        id: this.movieId,
        dayTime: time,
        showdate: date
      }

      this.places = await cardApi.getBookingPlaces(payload)
    },
    selectPlace(seat, row, isFree) {
      if (this.orderForm.seat === seat && this.orderForm.row === row) {
        this.orderForm.seat = ''
        this.orderForm.row = ''
        return
```

```
      }
      if (isFree) {
        this.orderForm.seat = seat
        this.orderForm.row = row
      }
    },
    async sendRequest() {
      if (this.orderForm.seat && this.orderForm.row) {
        await cardApi.bookingTicket(toSnakeCase(this.orderForm)).then(() => {
          this.showBookingForm = false
          this.showSuccessDialog = true
        })
      }
    },
    resetForm() {
      this.orderForm.row = null
      this.orderForm.seat = null
    }
  }
}
</script>

<style lang="scss">
@import "../assets/scss/mixins";
.card {
  text-align: left;

  &-wrap {
    position: relative;
  }

  &-image {
    height: 900px;
    @include _768{
      height: 400px;
    }
    @include _600{
      height: 300px;
    }
    img {
      width: 100%;
      height: 100%;
      object-fit: cover;
    }
  }

  &-wrap {
    display: flex;
    @include _1170{
      display: block;
    }
  }

  &-left {
    width: 40%;
    @include _1170{
      width: 100%;
    }
  }

  &-right {
    width: 60%;
    margin-left: 20px;
    @include _1170{
```

```
    width: 100%;
    margin-left: 0;
    margin-top: 50px;
  }
}

&-title {
  font-size: 32px;
  margin-bottom: 20px;
  font-weight: 700;
}

&-description {
  font-size: 18px;
  margin-bottom: 30px;
}

&-additional {
  font-size: 18px;
  font-weight: 300;
  .key{
    font-weight: 700;
    margin-bottom: 10px;
    margin-top: 10px;
  }
  a{
    color: black;
  }
}

&-sessions {
  margin-top: 50px;
  font-size: 20px;
  font-weight: 700;

  &__time {
    margin-top: 10px;
    @include _600{
      display: flex;
      flex-direction: column;
    }
    span {
      margin-right: 10px;
      cursor: pointer;
    }
  }

  ul {
    font-weight: 300;
    margin-top: 20px;
  }

  li {
    margin-bottom: 20px;
    display: flex;
    flex-direction: column;

    span {
      color: #7C818D;
    }
  }
}

&-form {
```

```
position: fixed;
margin: 0 auto;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
background-color: #fff;
width: calc(100% - 15px);
height: 800px;
box-shadow: 0px 0px 5px 4px rgba(0, 0, 0, 0.25);
width: 100%;
&__seats{
  @include _600{
    overflow-x: auto;
  }
}

&__close {
  position: absolute;
  top: 15px;
  right: 15px;
  cursor: pointer;
}

&__submit {
  margin-top: 50px;
  border: 1px solid #b3b3b3;
  width: 200px;
  padding: 20px 10px;
}

&__wrap {
  padding-top: 50px;
  height: 100%;
  display: flex;
  justify-content: center;
  align-items: flex-start;
  padding: 15px;
}

form {
  //width: 300px;

  @include _979{
    width: 100%;
    margin-top: 50px;
  }
  label {
    display: flex;
    flex-direction: column;
    margin-bottom: 30px;

    input {
      border: 1px solid #b3b3b3;
      height: 40px;
      padding-left: 30px;
      font-size: 18px;
    }
  }

  .row {
    display: flex;

    .seat {
      display: flex;
```

```
            border: 1px solid black;
            width: 20px;
            height: 20px;
            display: flex;
            justify-content: center;
            align-items: center;
            margin-bottom: 5px;
            font-size: 14px;
            cursor: pointer;
            background-color: red;
            @include _600{
              padding: 3px;
            }
            &.free {
              background-color: #fff;
            }

            &.selected {
              background-color: green;
            }

            &:not(:last-child) {
              margin-right: 2px;
            }
          }
        }
      }
    }

    &-success {
      position: fixed;
      margin: 0 auto;
      top: 50%;
      left: 50%;
      transform: translate(-50%, -50%);
      background-color: #fff;
      width: calc(100% - 150px);
      height: 400px;
      box-shadow: 0px 0px 5px 4px rgba(0, 0, 0, 0.25);
      display: flex;
      align-items: center;
      justify-content: center;

      p {
        font-size: 48px;
      }

      &__close {
        position: absolute;
        top: 15px;
        right: 15px;
        cursor: pointer;
      }
    }
  }
}
</style>
```

Папка views Catalog.vue

```
<template lang="pug">
  .catalog
    .container
      .catalog-search
        input(type="text" v-model="search")
      .catalog-sort
        .catalog-sort__type(v-for="(type,index) in sort()" :key="index"
@click="handleRequest(type.key,'')") {{type.name}}
      .catalog-container(v-if="list")
        product(v-for="movie in list" :key="movie.id" :movie="movie")
      .catalog-empty(v-else)
        | Empty data


</template>

<script>
import product from "@/components/CatalogItem.vue";
import apiCatalog from '@/api/catalog'

export default {
  name: "CatalogGuide",
  components: {
    product
  },
  data() {
    return {
      list: [],
      search: '',
      currentType: ''
    }
  },
  watch: {
    'search'() {
      this.handleRequest(this.currentType, this.search)
    }
  },
  async created() {
    await this.handleRequest(this.currentType, this.search)
  },
  methods: {
    async handleRequest(type, name) {
      this.list = await apiCatalog.getList(type, name)
    },
    sort() {
      return [
        {
          name: 'Екшн',
          key: '0'
        },
        {
          name: 'Пригоди',
          key: '1'
        },
        {
          name: 'Комедія',
          key: '2'
        },
        {
          name: 'Драма',
          key: '3'
        },
        {
          name: 'Хорор',
```

```
          key: '3'
        },
        {
          name: 'Вестерн',
          key: '4'
        },
      ]
    }
  }
}
</script>

<style lang="scss">
.catalog {
  &-container {
    display: flex;
    flex-wrap: wrap;
    width: 100%;
  }

  &-empty {
    margin-top: 25px;
    font-size: 26px;
  }

  &-search {
    input {
      border: 1px solid #b3b3b3;
      height: 30px;
      padding-left: 20px;
      font-size: 18px;
    }

    margin-bottom: 20px;
  }

  &-sort {
    display: flex;
    justify-content: center;

    &__type {
      cursor: pointer;
      font-size: 14px;
      font-weight: 700;

      &:not(:last-child) {
        margin-right: 10px;
      }
    }

    &__clear {
      border: 1px solid rgba(17, 17, 17, 0.15);
      margin-right: 25px;
      padding: 15px 20px;
      text-transform: capitalize;
    }

    .main-select {
      &:last-child {
        margin-left: 25px;
      }
    }
  }
```

```
}
</style>
```

App.vue

```
<template>
  <div id="app">
    <div class="wrapper">
      <router-view/>
    </div>
  </div>
</template>
<style src="@/assets/scss/main.scss" lang="scss"></style>
```

Main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'

Vue.config.productionTip = false

new Vue({
  router,
  render: h => h(App)

}).$mount('#app')
```

ДОДАТОК В
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

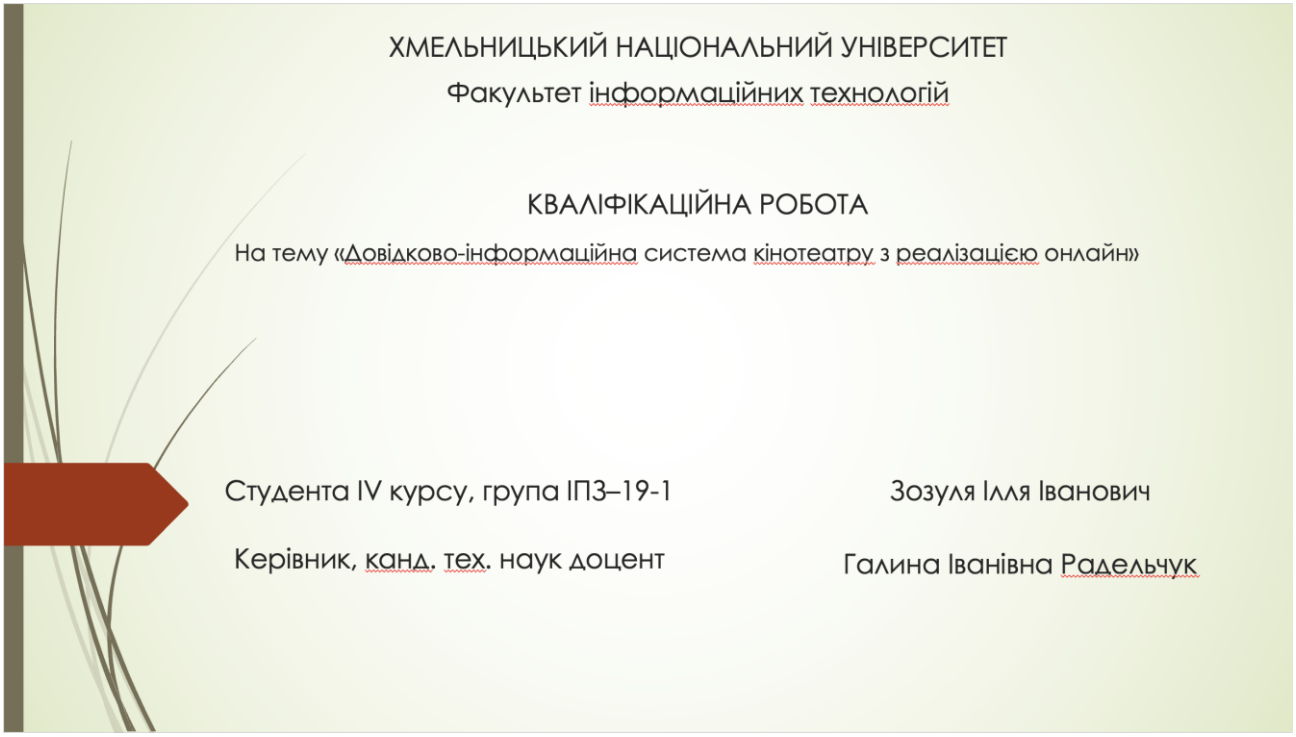


Рисунок В.1 – Титульний слайд

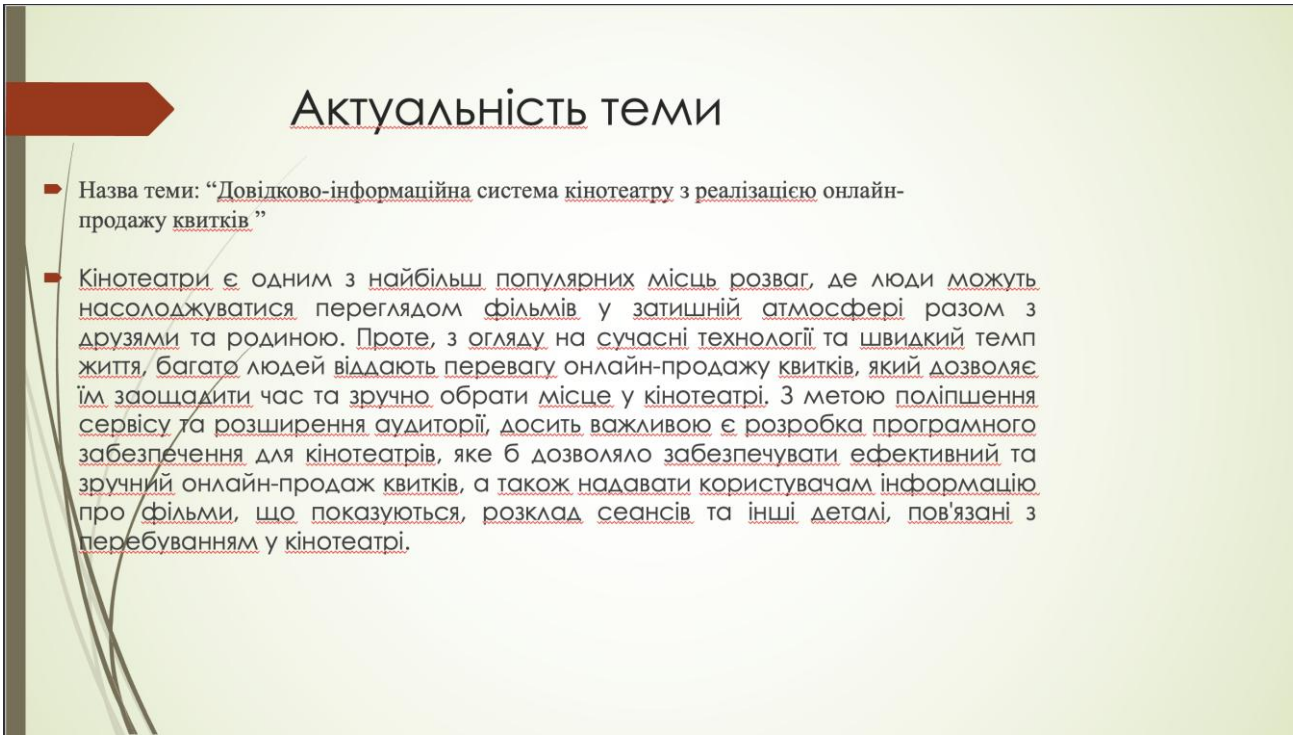


Рисунок В.2 – Актуальність теми

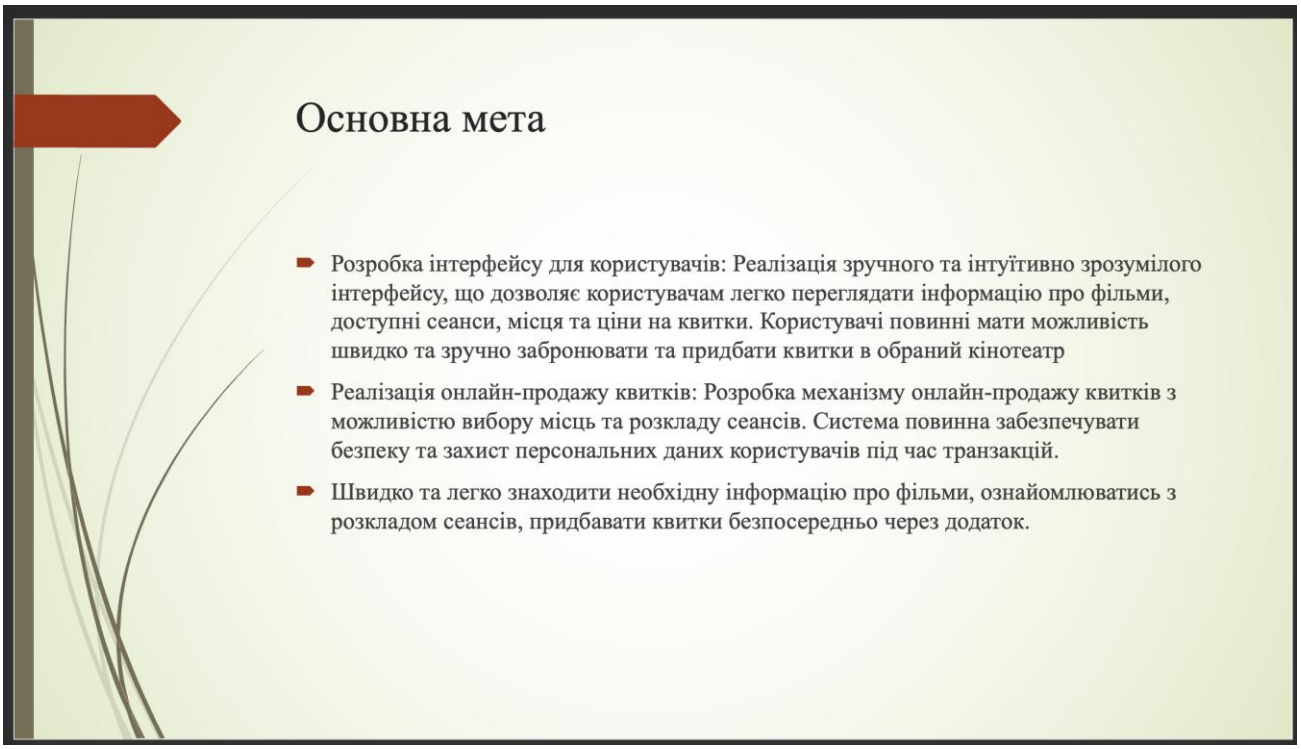


Рисунок В.3 – Основна мета

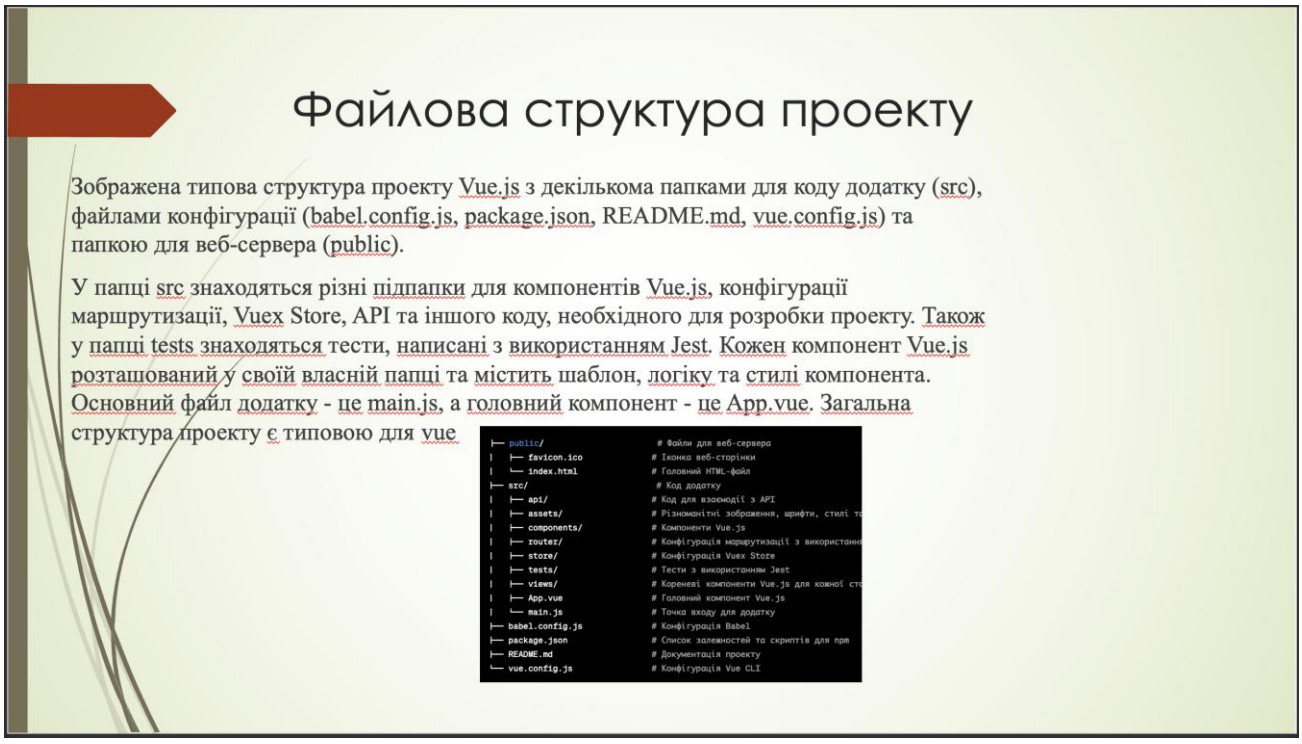


Рисунок В.4– Файлова структура

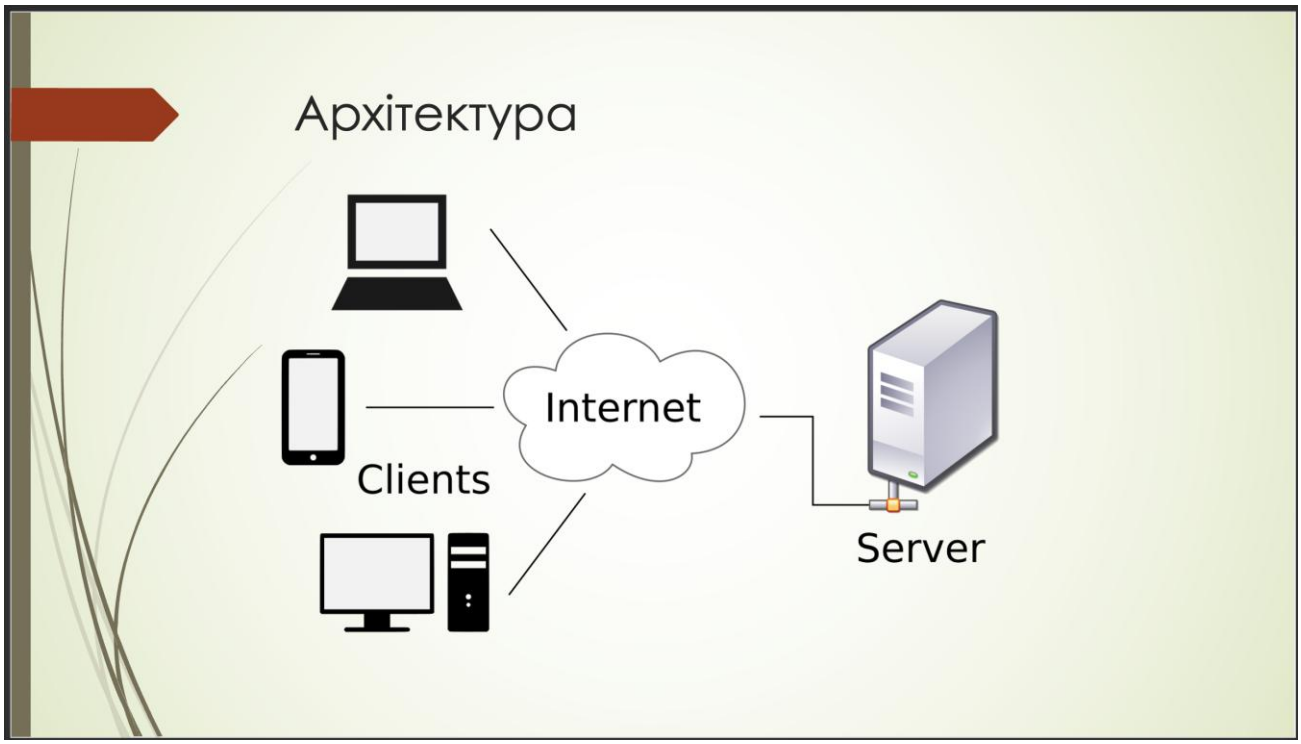


Рисунок В.5 - Архітектура

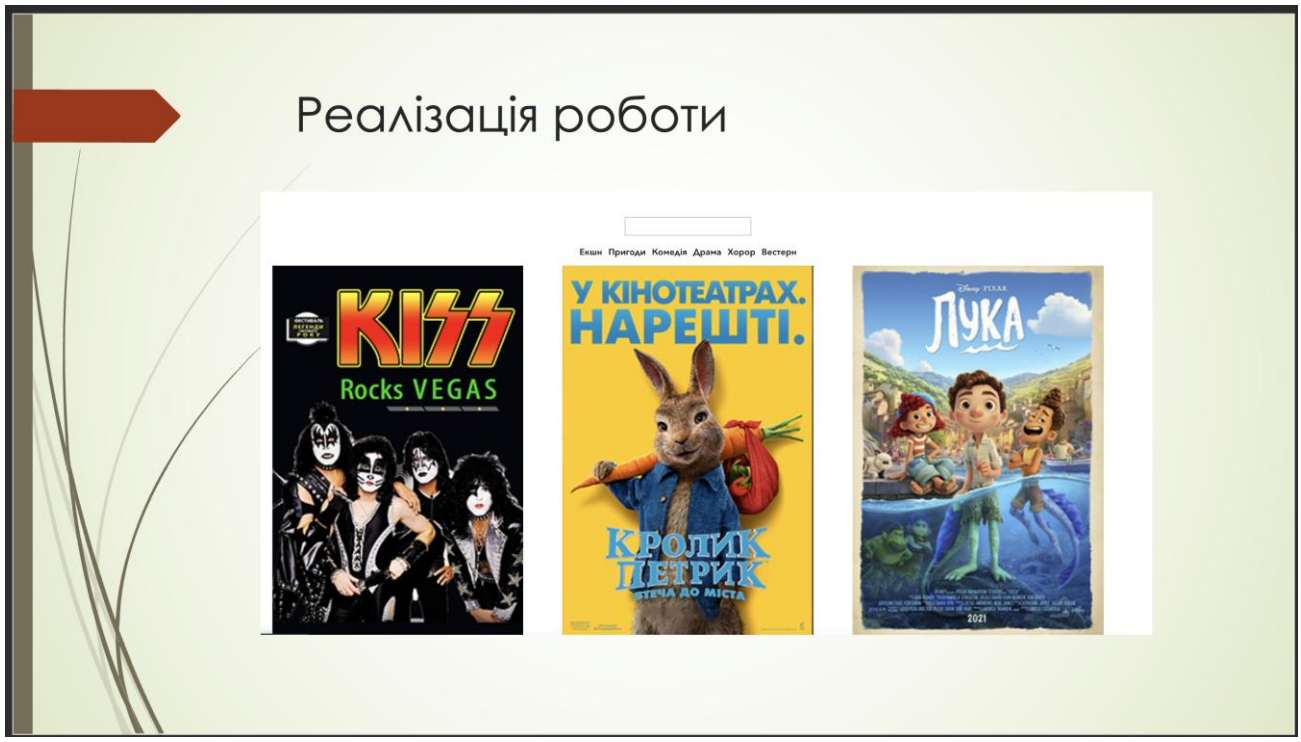


Рисунок В.6 – Реалізація роботи

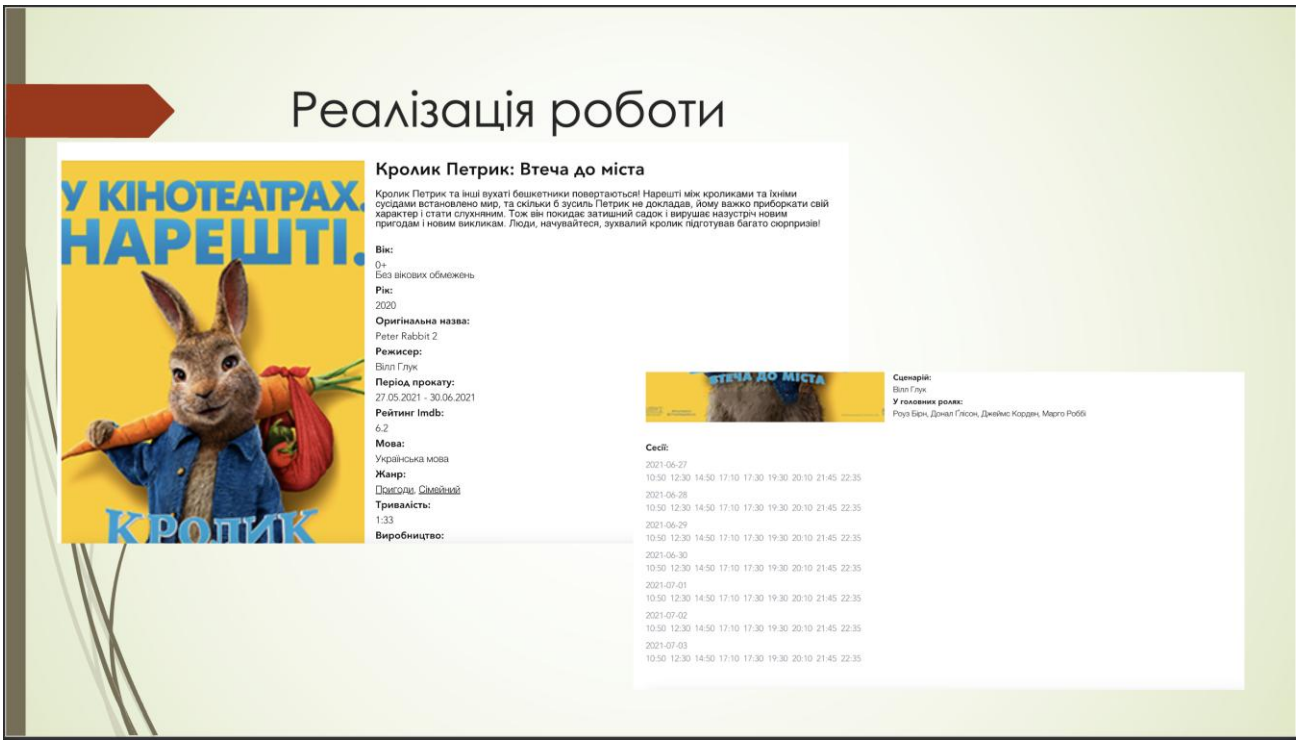


Рисунок В.7 – Реалізація 2

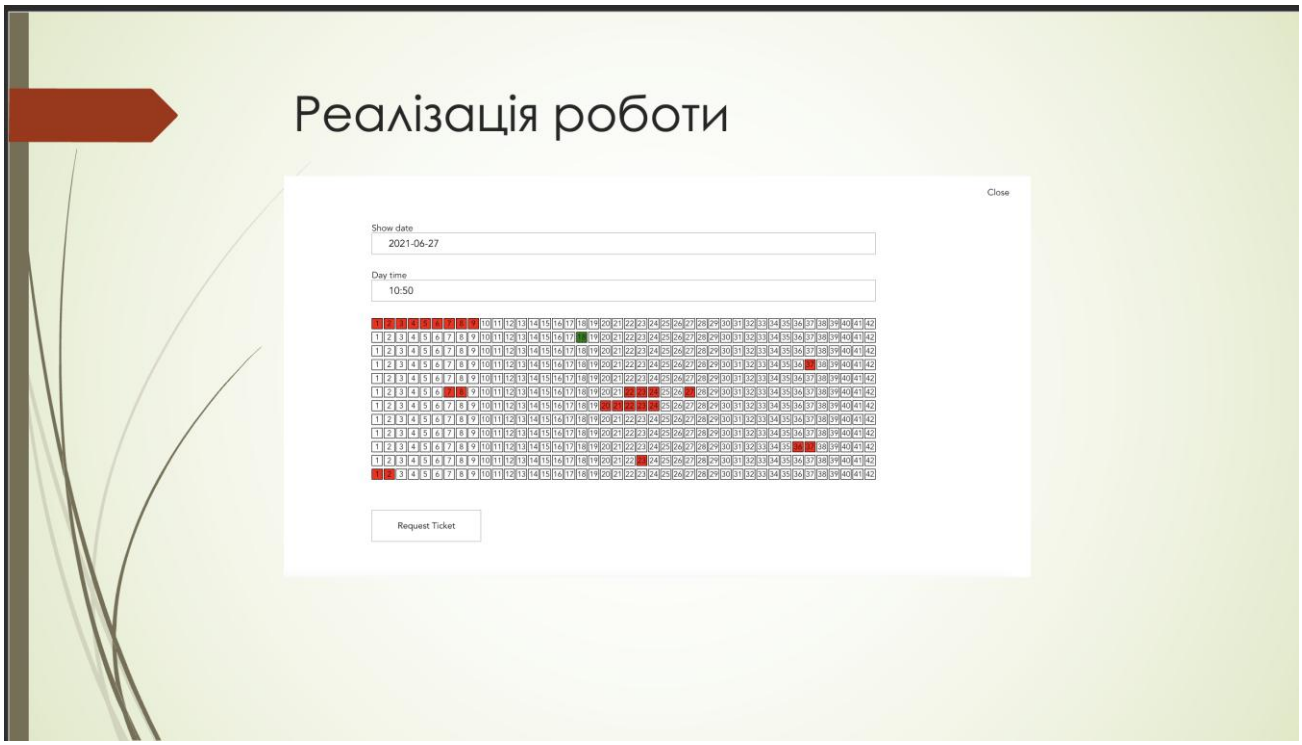
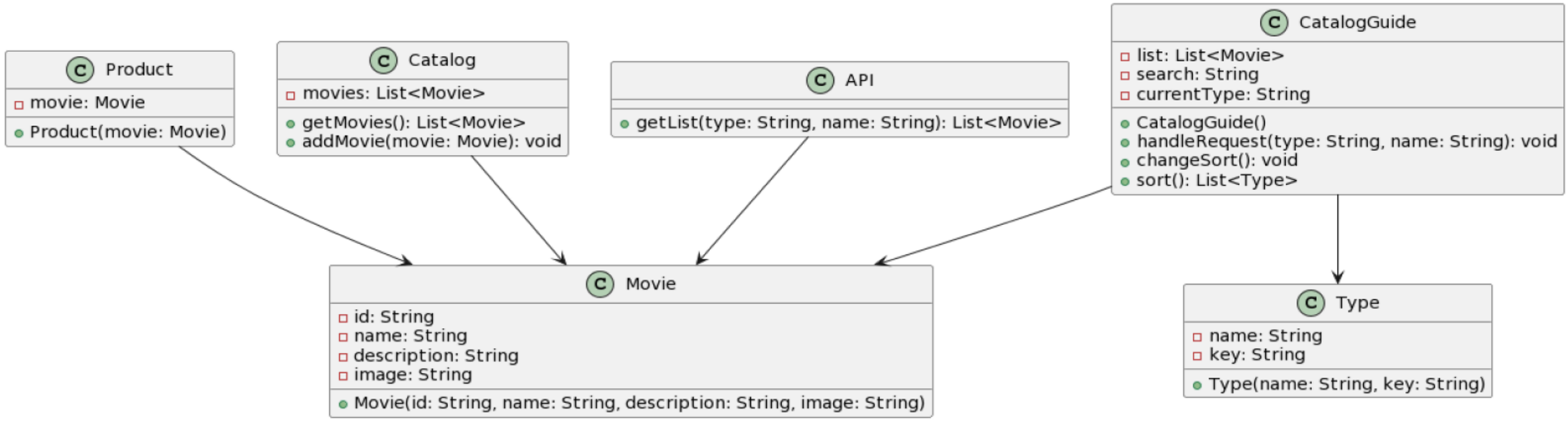
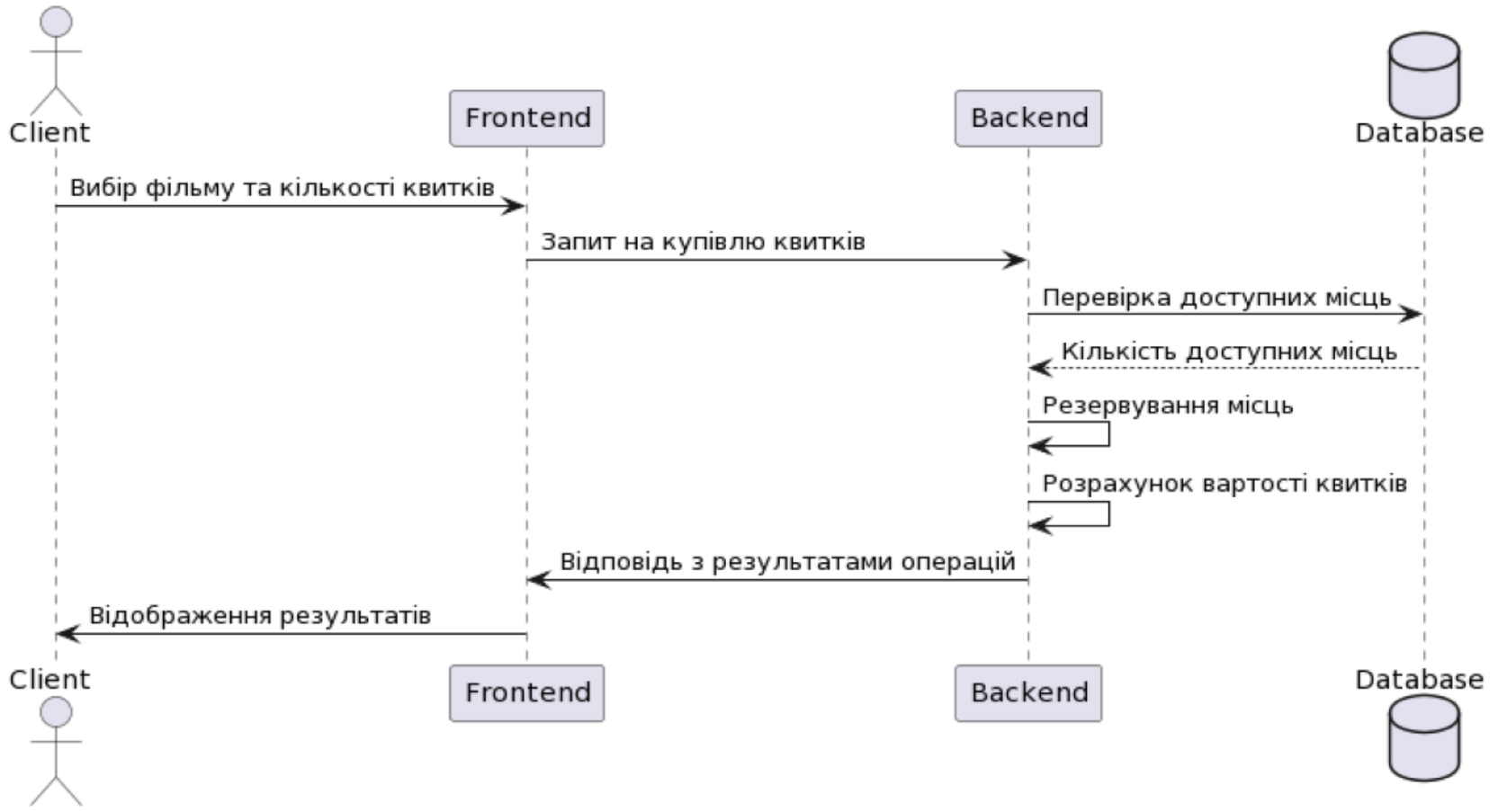


Рисунок В.8 – Реалізаця 3

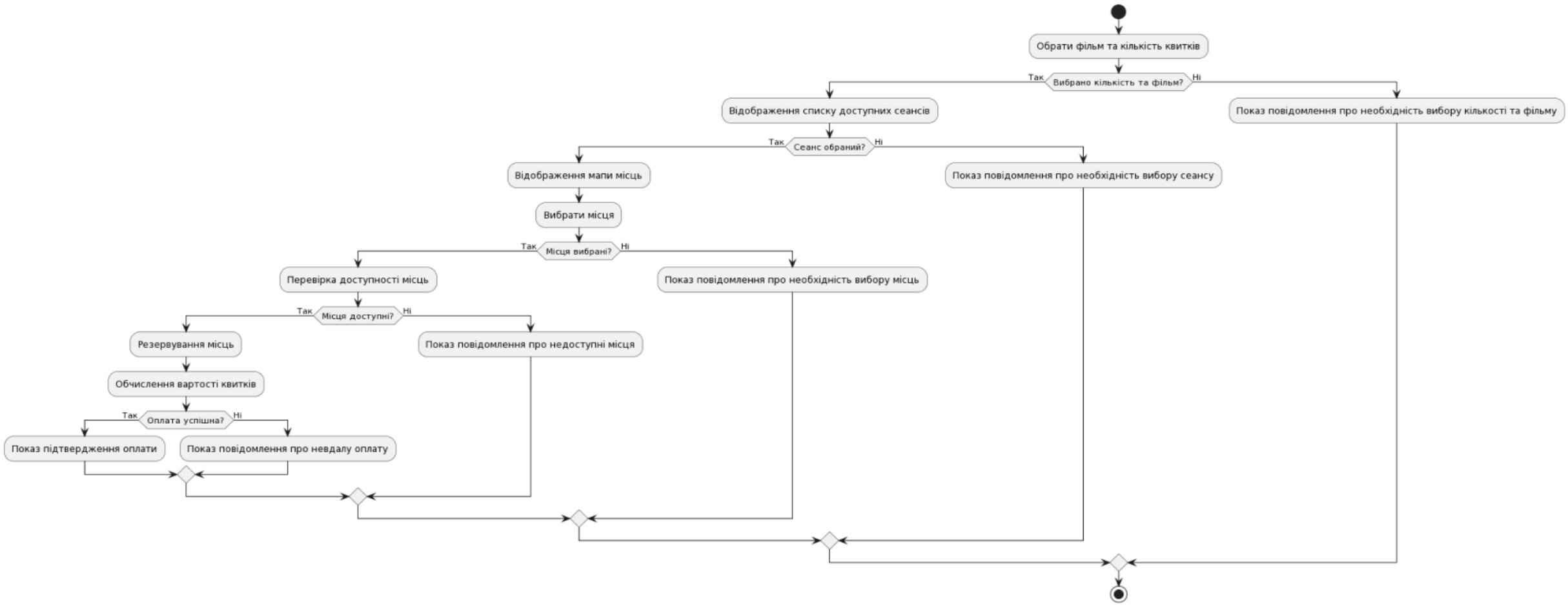
ГРАФІЧНА ЧАСТИНА



					КвРІП3.200137.01.09.Е8			
Зм.	Арк.	№докум.	Підпис	Дата	Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків UML-Діаграма класів	Літера	Маса	Масштаб
Розробив		Зозуля І.І.						
Керівник		Радельчук Г.І				Аркуш 1	Аркушів 3	
Консульт.								
Н.Контр.		Радельчук Г.І				ХНУ, ІП3-19-1		
Зав.каф.		Бедратюк Л.П						



					КвРІП3.200137.01.09.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків UML-Діаграма послідовності	Літера / Маса / Масштаб
Розробив		Зозуля І.І.				
Керівник		Радельчук Г.І				Аркуш 2 / Аркушів 3
Консульт.						
Н.Контр.		Радельчук Г.І				ХНУ, ІП3-19-1
Зав.каф.		Бедратюк Л.П				



					КвРІП3.200137.01.09.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків UML-Діаграма діяльності	Літера / Маса / Масштаб
Розробив		Зозуля І.І.				
Керівник		Радельчук Г.І				Аркуш 3 / Аркушів 3
Консульт.						
Н.Контр.		Радельчук Г.І				ХНУ, ІП3-19-1
Зав.каф.		Бедратюк Л.П				

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Зозулі І. І.
Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-19-1

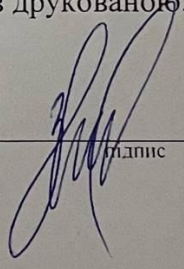
ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

14.06.2023
дата


підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірок: en_US, ru_RU, uk_UA. Помилок в документах: 13%

ID: 116625 Назва: БКР Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків Додано в БД: 2023-06-16 Автори: Зозуля І.І. Керівники: Радельчук Г.І. к.т.н. доц. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	84403	633	1753 (2%)	25 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015625627

Дата перевірки:
16.06.2023 13:00:14 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
16.06.2023 13:08:35 EEST

ID користувача:
100005589

Назва документа: **ДП_Зозуля(рефактор) для плагіату**

Кількість сторінок: **66** Кількість слів: **12417** Кількість символів: **97766** Розмір файлу: **18.47 MB** ID файлу: **101527240**

6.62%
Схожість

Найбільша схожість: **1.36%** з джерелом з Бібліотеки (ID файлу: **1015074274**)

5.85% Джерела з Інтернету	909	Сторінка 68
2.3% Джерела з Бібліотеки	91	Сторінка 73

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Зозуля Ілля Іванович.

Тема Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 66

1. Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі проведена робота над розробкою довідково-інформаційної системи для кінотеатру з реалізацією онлайн-продажу квитків. Були проаналізовані існуючі застосунки та їхні функціональні можливості, а також виявлені недоліки та проблеми, що можуть бути враховані при розробці нової системи. Для реалізації проекту було обрано фреймворк Vue.js, оскільки він забезпечує швидку та ефективну розробку користувацького інтерфейсу, а також надає зручні інструменти для керування станом додатку та навігацією між сторінками.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням вимог стандартів.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів

У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені вимоги до розроблюваної довідково-інформаційної системи кінотеатру, виконана постановка задачі. У другому розділі обрана архітектура системи, спроектована її структура та база даних, а також проведено аналіз та вибір технологій і засобів реалізації системи. У третьому розділі описана реалізація системи та її тестування.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною. Також для проектування та реалізації системи застосовано новітні технології та актуальні архітектурні рішення.

5. Негативні сторони роботи Робота дещо переобтяжена описовим матеріалом.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана на достатньому рівні, відповідає поставленій задачі та заслуговує на оцінку «задовільно».

РЕЦЕНЗЕНТ

Лисенко Сергій Миколайович, професор кафедри КІТС ХНУ

19.06.2023 р.

(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів подібності щодо роботи, продукованими програмно-технічним засобом (ами) перевірки текстів на плагіат:

Назва: «Довідково-інформаційна система кінотеатру з реалізацією онлайн-продажу квитків»
Автор: Зозуля Ілля Іванович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Радельчук Галина Іванівна, кандидат технічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) в якості запозичень системою Unicheck було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 1.0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 6.62% і адресується до 909 джерел з інтернету і 91 джерела з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 16.06.2023 р.

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Галина РАДЕЛЬЧУК