Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка клієнт-серверної системи мережі кінотеатрів «Metropolitan» на платформі Loravel»

Студента 4 курсу, 7 групи,
спеціальності 121 «Інженерія програмного забезпечення» _____ Ліщука Богдана Валерійовича
підпис студента

Науковий керівник
кандидат технічних наук,
доцент _____ Савченко Тетяна Віталіївна
підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент _____ Цензура Микола Олександрович
підпис керівника

КИЇВ – 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь — бакалавр
Спеціальність 121, Інженерія програмного забезпечення

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«7» листопада 2019 р.

Завдання
на випускну кваліфікаційну роботу студентові

Ліщуку Богданові Валерійовичу

1. Тема випускної кваліфікаційної роботи: «Розробка клієнт-серверної системи мережі кінотеатрів «Metropolitan» на платформі Loravel».

Затверджена наказом ректора від «02» грудня 2019 р. № 4112

2. Строк здачі студентом закінченої роботи _____
_____

3. Цільова установка та вихідні дані до роботи:

Мета роботи: розробка клієнт серверної системи на базі лоравель для спрощення способу придбання квитків у мережі кінотеатрів “Метрополітан”.

Об’єкт дослідження: мережа кінотеатрів “Метрополітан”.

Предмет дослідження: клієнт-серверна система на базі лоравель.

Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

4. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

Вступ

Розділ 1 WEB-СЕРВІСИ ПО ПРИДБАННЮ КВИТКІВ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні відомості про web-сервіси по придбанню квитків

1.2 Технічне завдання

1.3 Висновки до розділу 1

Розділ 2 АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ НА БАЗІ ЛОРАВЕЛЬ

2.1 Загальні відомості про програмне забезпечення клієнт-серверної системи

2.2 Детальний огляд фреймворку Laravel

2.3 Розробка дизайну

2.4 Розробка бази даних

2.5 Висновки до розділу 2

Розділ 3 Створення програмного продукта

3.1 Реалізація веб-системи

3.2 Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	Вибір теми випускної кваліфікаційної роботи		
2.	Вступ та перелік літературних джерел		
3.	Розділ 1. Технічне завдання та дослідження необхідних ресурсів для створення програми «інтернет магазин з продажу комп'ютерної техніки»		
4.	Розділ 2. Постановка задачі та моделювання архітектури необхідних ресурсів для інтернет магазину з продажу комп'ютерної техніки		
5.	Розділ 3. Реалізація веб-додатку «інтернет-магазин з продажу комп'ютерної техніки»		
6.	Висновки		
7.	Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)		
8.	Підготовка автореферату та презентації доповіді		
9.	Попередній захист випускної кваліфікаційної роботи		
10.	Зовнішнє рецензування випускної кваліфікаційної роботи		
11.	Здача прошитої випускної кваліфікаційної роботи на кафедру		
12.	Публічний захист випускної кваліфікаційної роботи		

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи Савченко Т. В.

9. Гарант освітньої програми Цензура М. О.

10. Завдання прийняв до виконання студент Ліщук Б. В.

11. Відгук керівника випускної кваліфікаційної роботи

Науковий керівник випускної кваліфікаційної роботи _____

(підпис, дата)

Відмітка про попередній захист Савченко Т. В. _____

(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента Ліщука Б. В. може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М. О.

Завідувач кафедри Криворучко О. В.

«_____»_____20 ___ р.

ЗМІСТ

					КНТЕУ 121 07-09.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*			
Зав. кафедри		*Криворучко О.В.*			*Розробка клієнт-серверної системи мережі кінотеатрів «Metropolitan» на платформі Loravel*	*Стадія*	*Аркуш*	*Аркушів*
Керівник		*Савченко Т.В..*				*Зміст*	*2*	*41*
Гарант		*Цензура М.О.*						
Розроб.		*Ліщук Б.В.*			*Зміст*	*Факультет інформаційних технологій, 4 курс, 7 група*		

ВСТУП

Кількість технологій з кожним роком росте зі швидкістю геометричної прогресії. Ще 20 років тому в найвіддаленіших куточках України люди не знали що таке мобільний зв'язок і контактували один з одним з-за допомогою листування, то зараз в тих же населених пунктах появилась можливість не тільки спілкування по відео зв'язку, але й цілий спектр послуг, якими можна скористуватись з-за допомогою інтернету та електротехніки. Прогрес людства в області техніки та технологій неможливо переоцінити. Ми настільки стали зв'язані з ними, що різке щезнення хоча б одного елементу приведе до низки економічних та соціальних криз.

Головна мета усіх інновацій це зробити життя людей простішим. В даний час, для нас такі речі, як придбання побутової техніки, їжі, білетів і навіть домашніх тварин є звичними, як колись було звичним стояти в чергах до кас. Практично всі сфери послуг можна замовити через інтернет, що також стосується сфери розваг.

Кінематографія є однією з найважливіших та найприбутковіших сфер у світі. Майже у кожної людини, як мінімум є улюблений фільм, серіал або ж актор.

У 1895 році, задовго до появи масового електронного телебачення, був проведений перший кіносеанс, а через 15 років кількість кінотеатрів виросла до 6000 тисяч. Масова популяризація кінопрокату призвела до створення всім відомого Голлівуду та десятків кінокомпаній. Це все призвело до збільшення кількості фільмів, які відповідно потрібно збувати на ринку.

Кількість кінотеатрів в даний час неможливо підрахувати, адже їх кількість збільшується на десятки тисяч в нових містах з кожним роком

Цифрова еволюція вплинула не тільки на розвиток кінематографу, а й способи збування продукту. Замість довгих черг, тепер можна в декілька натисків придбати білети у кіно. Автоматизація та спрощення цього процесу є такою ж невід'ємною та важливою частиною в кінематографі, як і якість картинки, яку ми спостерігаємо на екранах. Кожен крок який споживач робить має бути інтуїтивно зрозумілим. Від цього залежить рейтинг кінопрокатної компанії та лояльність її клієнтів. Тому, обрана мною тема «Розробка клієнт серверної системи мережі кінотеатрів "Метрополітан" на базі лоравель» є актуальною.

Метою дослідження випускної кваліфікаційної роботи є розробка клієнт серверної системи на базі лоравель для спрощення способу придбання квитків у мережі кінотеатрів "Метрополітан".

Об'єктом дослідження є мережа кінотеатрів "Метрополітан".

Предметом дослідження є клієнт серверна система на базі лоравель .

Завдання дослідження:

- Створити дизайн веб-системи;
- розглянути інструменти розробки, з-за допомогою яких буде створена веб-система;
- спроектувати логічну та фізичну моделі на основі яких створити базу даних;
- створити клієнт серверну систему.

Методи дослідження: В проекті використовуються методи об'єктно орієнтованого програмування, програмування на мовах високого рівня PHP та Javascript, розробка інтерфейсу користувача.

РОЗДІЛ .1
WEB-СЕРВІСИ ПО ПРИДБАННЮ КВИТКІВ ТА ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні відомості про web-сервіси по придбанню квитків

Усі існуючі мережі кінотеатрів пропонують своїм клієнтам спосіб придбання квитка через мережу інтернет. Відповідно конкуренція компаній проявляється у зручності використання сервісів. Здебільшого усі кінотеатри працюють по одній й тій самій системі. У всіх кінотеатрах є зали, сеанси і список актуальних та новий фільмів на вибір звичайно. Відповідно веб-сервіси кінопрокатних компаній пропонують вибір фільм, час та місце в залі. Усі ці особливості є однаковими для кожного веб-сайту по придбанню квитків у кінотеатр. Тому в більшості основна конкуренція web-сервісів проявляється в дизайнів. Частина веб-сайтів пістрявить яскравими кольорами і великою кількістю малюнків, фотографій та відео. Інша ж частина схильна до делікатних та рівних ліній та спокійних кольорів. В такому тихому стилі домінують всього декілька кольорів і за яскравість на сайті можуть відповідати тільки різнокольорові обкладинки фільмів. До такого дизайну належить розроблений компанією Google “Material design”, принципи якого використовується не тільки у веб-застосунках а й для інтерфейсів операційної системи Android.

Відступаючи від теми дизайну для веб-застосунків, потрібно затронути також внутрішню кухню веб-сайту, а саме фреймворк лоравель, на базі якого буде створюватись веб-сервіс. PHP-фреймворк пропонує низку інструментів для

побудови цілісного веб-додатку. З допомогою лоравель ми зможемо легко не витрачати лишні куски коду.

Також потрібно згадити про бази даних, без яких неможливо побудувати ні одну систему світі технологій. Зберігати та накопичувати велику кількість інформації дозволяють бази даних. Для легшого та гнучкішого управління цими даними використовуються СУБД – системи управління базами даних.

1.2 Технічне завдання

1. Загальні відомості
   1.1. Найменування системи – система мережі кінотеатрів “Метрополітан” на базі лоравель.
      1.1.1. Повне найменування системи – клієнт серверна система мережі кінотеатрів “Метрополітан” на базі лоравель
      1.1.2. Скорочене найменування системи – система мережі кінотеатрів “Метрополітан”
   1.2. Планові терміни початку та закінчення робіт
      1.2.1. Початок робіт – 01.03.2020
      1.2.2. Закінчення робіт – 01.05.2020
   1.3. Порядок та пред’явлення результатів робіт – пред’явити робочу систему керівнику
   1.4. Головний бенефіціар та потенційні користувачі системи – мережа кінотеатрів “Метрополітан” та клієнти і працівники компаніїї.
2. Мета та призначення створення системи
   2.1. Призначення системи – бронювання квитків на сеанси фільмів онлайн
   2.2. Мета створення системи – створити систему для бронювання квитків на сеанси фільмів
3. Вимоги до системи

3.1. Вимоги до системи в цілому – у системі повинна бути реалізована можливість бронювання квитків на сеанси фільмів, а також адміністрування самої системи.

3.1.1. Вимоги до структури та функціонування системи, перелік підсистем – структура базується на архітектурній моделі MVC(Model, View, Controller).

3.1.1.1. Вимоги до способів і засобів інформаційного обміну між компонентами системи – зв’язок між компонентами системи повинно один із елементів MVC моделі, а саме Controller.

3.1.1.2. Вимоги до режимів функціонування системи – система повинна функціонувати у режимі онлайн.

3.1.1.3. Вимоги до діагностування системи – система повинна працювати безперебійно та успішно виконувати усі процеси.

3.1.1.4. Вимоги до режимів управління системою – повинно бути реалізовано два режими, а саме гостьовий і адміністраторський

3.1.2. Показники призначення

3.1.2.1. Параметри, що характеризують ступінь відповідності системи призначенням – безперебійність роботи системи, відсутність помилок.

3.1.2.2. Вимоги до пристосованості системи до змін – в систему повинно інтегрована можливість прибдання квитка онлайн.

3.1.2.3. Вимоги до збереження працездатності системи в різних ймовірних умовах – система повинна бути працездатна тільки в онлайн режимі.

3.1.3. Вимоги до надійності

3.1.3.1. Склад показників надійності системи – система повинна мати SSL сертифікат.

3.1.3.2. Вимоги до надійності технічних засобів і програмного забезпечення – програмне забезпечення повинне відповідати усім стандартам надійності CORS(Cross-Origin Resource Sharing).

3.1.3.3. Вимоги до методів оцінки і контролю показників надійності на різних стадіях створення системи – на стадії тестування дозволена поява помилок, на стадії здачі системи не дозволена поява помилок.

3.1.4. Вимоги до ергономіки та технічної естетики – зі знарядь праці необхідний комп'ютер на робоче місце.

3.1.5. Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи – компоненти системи повинні зберігатися на сервері.

3.1.6. Вимоги до захисту інформації від несанкціонованого доступу – доступ до інформації надається тільки адміністраторам у вигляді логіну та паролю.

3.1.6.1. Вимоги до інформаційної безпеки – закрити усім користувачам веб-сервісу, окрім адміністраторів, доступ до адмін панелі.

3.1.6.2. Вимоги до антивірусного захисту – антивірусна програма повинна бути стабільною та надійною.

3.1.6.3. Розмежування відповідальності ролей при доступі – повинно існувати два типи ролей, а саме: гість та адміністратор.

3.1.7. Вимоги до захисту від впливу зовнішніх факторів

3.1.8. Вимоги безпеки – система повинна бути безпечною для її користувачів.

3.2. Перелік підсистем системи – підсистеми відсутні.

3.3. Вимоги до видів забезпечення

3.3.1. Вимоги до математичного забезпечення – у системі не буде використовуватись математичні вичислення.

3.3.2. Вимоги до інформаційного забезпечення – інформаційне забезпечення повинне бути цілісним і захищеним від несанкціонованого доступу.

3.3.2.1. Вимоги до складу, структури і способів організації даних в системі – усі дані повинні зберігатись з-за допомогою баз даних і управлятись СУБД.

3.3.2.2. Вимоги до інформаційного обміну між компонентами системи - передача інформації між компонентами системи має виконуватись стандартними протоколами на рівні програмного забезпечення або на рівні платформи (системи керування БД, веб-серверів тощо).

3.3.2.3. Вимоги до інформаційної сумісності із суміжними системами – система не повинна бути закрити від інтеграції із суміжними системами.

3.3.2.4. Вимоги використання класифікаторів та уніфікованих документів – не будуть використовуватись.

3.3.2.5. Вимоги щодо застосування систем управління базами даних – система управління базами даних повинна бути оснащена програмними засобами, необхідними для створення, використання та підримки баз даних.

3.3.2.6. Вимоги до структури процесу збору, обробки, передачі даних в системі представлення даних – процес збору, оброки та передачі даних в системі повинен бути захищеним, безпечним та швидким.

3.3.2.7. Вимоги до захисту даних від руйнувань при аваріях і збоях в електроживленні системи – доступ до даних повинен бути заборонений при аваріях та збоях в електроживленні системи.

3.3.2.8. Вимоги до контролю, зберіганню, оновленню та відновленню даних – дані повинні зберігатись на сервері, доступ до контролю, зберігання, оновленню та відновленню повинен проходити тільки по захищених канал та повинні використовуватись безпечні протоколи.

3.3.2.9. Вимоги до процедури надання юридичної сили документам, що продукуються технічними засобами системи – не передбачуються.

4. З-за допомогою програмного забезпечення, клієнти повинні мати можливість забронювати місце у мережі кінотеатрів “Метрополітан”. Клієнт повинен мати змогу вибрати фільм, сеанс та місце в залі. Забронюване місце в залі, вибраний сеанс та назва фільму повинні приходити на пошту електронним повідомленням. Адміністратор веб-системи повинен мати змогу додавати фільми, режисерів та жанри, створювати сеанси і додавити зали. Усі дії повинні здійснюватись з-за допомогою мережі інтернет. Програмне забезпечення повинно базуватись на використанні:

4.1. Apache HTTP сервер

4.2. Мінімальна версія мови програмування PHP - 7.2

4.3. Фреймворк Laravel для PHP

4.4. Система керування базою даних Mysql

4.5. Система адміністрування систем керування баз даних – HeidiSQL

4.6. Бібліотека для javascript - Jquery

5. Для технічного забезпечення рекомендовано використовувати комп'ютер з такими характеристиками:

5.1. Windows 7,8,10

5.2. Процесор Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz

5.3. 8 гігабайт оперативної пам'яті

5.4. 64-розрядна операційна система на базі процесора x64

6. Вимоги до методичного забезпечення – повинна бути створена методичний конспект по використанню адмін панелі. В конспекті повинні бути описані покроково способи адмінстрування веб-сервісу, а саме:

6.1. Додавання фільму

6.2. Додавання жанрів

6.3. Додавання режисерів

6.4. Додавання залів

6.5. Додавання сеансів

1.3 Висновки до розділу 1

У першому розділі було проаналізовані загальні відомості про веб-сервіси по придбанню квитків до кінотеатрів. Також було створено та узагальнено технічне завдання.

РОЗДІЛ 2.
АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ НА БАЗІ ЛОРАВЕЛЬ

2.1 Загальні відомості про програмне забезпечення клієнт-серверної системи

Клієнт серверна модель визначається розподілом дій між клієнтом та сервером. Існує два модуля – клієнтський та серверний, які взамодіють між собою шляхом передачі даних. Веб-сервер відповідає за прийом HTTP-запитів від клієнтів та видає їм відповіді у зазвичай вигляді HTML сторінок, на яких розміщені тексти, фотографії, відео або аудіо, тощо.

Розрізняють два основних типа веб-сайтів – це статичний та динамічний. Під статичним типом розуміється веб-сайт, який складається з незмінних html-сторінок. Під динамічним типом розуміється же навпаки сайт, де контент html-сторінок може змінюватись при певних сценаріях, якими слідує користувач. Найпростіший приклад, це залишений клієнтом коментар під статею веб-блога. Для створення таких сайтів використовуються серверні мови програмування PHP, Java або ж Ruby on rails та додатково база даних. Коли користувач переходить по посилання, веб-браузер робить запит на веб-сторінку, яка знаходиться за адресою цього посилання. Веб-сервер отримавши цей запит передає необхідну сторінку інтерпретатору PHP, який в свою чергу виконує написані сценарії в файлі. Це можуть бути запити до бази даних, різні обчислення або ж відправка повідомлення на пошту. Після виконаної роботи, веб-сервер повертає згенеровану з-за допомогою PHP html-сторінку клієнту. Простим прикладом таких динамічних

веб-сторінок є блог. Автор блогу може створювати та зберігати свої статті. І для зручного користування особистим блогом, були придумані та створені адмін панелі, доступ до яких можна отримати зазвичай тільки з-за допомогою логіну та паролю. Згодом найпростіші види адмін панелей переросли в великі CMS(Системи керування змістом). Wodpress, OpenCart, Joomla є найпопулярнішими, як називають їх у професійному середовищі, двигунами у світі. У них уже є вбудовані адмін панелі, базово налаштована база даних та пропонують набір інструментів з якими можна створити доволі хороші та зручні веб-сайти. Усі ці системи є відкритим програмним середовищем, що означає високу гнучкість в розробці. Більше того, висока популярність цих CMS призвела до маштабного розвитку, тому в даний час з-за допомогою систем керування змістом можна створити не тільки звичайні блоги, але й високої складності веб-системи, як наприклад онлайн-магазини. Але бувають випадки, коли гнучкості CMS не вистачає і потрібно рухатись трішки іншим способом, а саме розробляти усе з нуля.

PHP – це серверна мова програмування. Приставка “серверна” використовується тому що скрипти написані цією мовою виконуються на віддаленому сервері. Користувачу зовсім не потрібно встановлювати PHP собі на комп’ютер для того щоб контент динамічно змінювався у веб-браузері. .З-за допомогою цієї серверної мови можна написати будь-якої складності веб-систему. Недарма такі відомі веб-сайти як Facebook та Вікіпедія написані на саме PHP. Але важливо підкреслити, що використовуючи тільки PHP, без додаткових інструментів веб-система буде створюватись дуже довго і буде занадто важкою. Тому, щоб вирішити ці стандартні проблеми розробки, було винайдені фреймворки. На даний момент, якщо створюється веб-система, то завжди використовується фреймворк. Для кожної мови програмування існують різні фреймворки. Зазвичай вони дуже схожі, але завжди є відмінності. Ларавель,

симфоні, зенд, юай2 є найбільш популярними фреймворками. Функціональність їх дуже обширна, тому кожен розробник вибирає підходящий фреймворк на особистий смак та аромат. У нашому випадку буде використаний фреймворк ларавель.

Ларавель – це безкоштовний PHP фреймворк, створений у 2011 році Тейлором Отвелом. Фреймворк використовує за основу архітектурний шаблон MVC, тому відразу зрозуміло, що використання контроллерів та моделей присутні. Ларавель першим приходить на думку для більшості розробників, які чують вираз PHP фреймворк. Він є одним з найпопулярніших серед PHP. І не дарма, адже ларавель має велику кількість ключових особливостей. Перше, що кидається в очі це чудовим синтаксисом. Зрозуміти, що виконує певний рядок коду без заглядання в документацію не викликає великих труднощів. Синтаксис є доволі простим та елегантним. Друге, що дає перевагу над іншими фреймворками – це функціональність. Базової функціональності вистачає для створення середнього веб-додатку. До того ж, фреймворк прекрасно інтегрується з додатковими бібліотеками та платформами, що дає можливість створювати високомасштабні проєкти. Чималий вплив на популярність ларавель мають встроєні функції. Одна авторизація та аутентифікація дуже багато коштує. Підключити такий модуль можна з-за допомогою одного рядку коду, що зберігає велику кількість часу для розробки. Також варто згадати про маршрутиризацію, кешування та інші. Також дуже важливо підкреслити, що велика популярність призвела до появи обширної кількості уроків, гайдів та навчальних відео. Вирішення практично 90% проблем зв'язаних з розробкою веб-застосунку з-за допомогою ларавель можна знайти на спеціалізованих форумах. Така велике використання та підтримка розробниками фреймворку відбивається на розвитку фреймворку. Обновлення виходять практично кожного місяця, а масштабне обновлення з нижчою на вищу версію виходить кожного року. Також варто

підкреслити чудово написану документацію. Вона охоплює практично усі закутки розробки, від встановлення до вигрузки готової веб-системи в мережу інтернет.

2.2 Детальний огляд фреймворку Laravel

Laravel – це безкоштовний PHP фреймворк загального призначення з відкритим кодом, який з'явився на світ порівняно недавно – в 2011 році, але, завдяки стрімким темпам розвитку і величезної армії шанувальників, сьогодні він є одним з найпопулярніших PHP двигунів. Самі творці Laravel назвали його «framework for artisans», що в перекладі означає «фреймворк для ремісників», натякаючи на те, що дана платформа дає розробникам повну свободу творчості, не створюючи перед ними жодних перешкод в процесі розробки. Уже в кінці 2013 року Laravel мав версію 4.1 і був названий «найбільш багатообіцяючим проєктом на 2014 рік» за версією sitepoint.com. А в 2015 і 2016 роках він був визнаний найпопулярнішим PHP фреймворком за версією того ж видання – sitepoint.com, яке щорічно проводить опитування серед тисяч розробників по всьому світу.

Інформативна документація – особливість Laravel, з якої неминуче стикаються всі розробники при освоєнні нової технології. Це документація Laravel, яка є дуже хорошою і структурованою. Це також додає популярності даного двигуну серед розробників. У Laravel документації кожній конструкції і процесу присвячена окрема стаття. Оскільки у даного PHP framework маса послідовників по всьому світу, то в мережі можна знайти безліч різних спільнот і призначених для користувача перекладів статей.

MVC структура коду Laravel framework відповідає популярному паттерну проектування MVC, тобто в ньому можна виділити моделі (models), уявлення (views) і контролери (controllers). Даний шаблон проектування зарекомендував себе як перевірене часом рішення ефективної структури додатків, в першу чергу, в вебі, що дозволяє відокремити логіку додатки від його візуальної частини. MVC

дозволяє робити код більш читабельним, а процес розробки комфортним, розмежовуючи роботу frontend- і backend-розробників.

Artisan - це консоль Laravel, в арсеналі команд якої є робота з міграціями, контролерами і моделями, авторизацією і іншими базовими компонентами фреймворка. Ключ artisan використовується у кожній консольній команді від фреймворку.

Для роботи з базою даних існують міграції. Це свого роду, контроль версій для структури таблиць БД. Кожен файл міграції містить або структуру таблиць, або зміни її структури. Тобто процес створення нових таблиць, зв'язків або будь-яких сутностей БД в Laravel фреймворку відбувається з-за допомогою створення міграції і запуск її за допомогою спеціальних консольних команд artisan.

Приклад створення таблиці користувачів сайту в відповідної міграції:

```
Schema :: create ( 'users', function (Blueprint $ table) {
$ Table-> increments ( 'id');
$ Table-> string ( 'name');
$ Table-> string ( 'email') -> unique ();
$ Table-> string ( 'password');
$ Table-> rememberToken ();
$ Table-> timestamps ();
});
```

Також в двигуні є безліч методів для маніпуляцій міграціями: відкат окремих міграцій або ж скидання всіх і т.д.

Blade - це власний шаблонізатор з набором своїх директив. На відміну від інших популярних двигунів шаблонів PHP, Blade не обмежує використання звичайного PHP-коду у своїх уявленнях. Насправді всі уявлення Blade складаються у звичайний PHP-код і зберігаються в кешованому стані, поки вони не будуть змінені, тобто Blade додає по суті нульові накладні витрати до вашої програми. Файли перегляду blade використовують розширення файлу .blade.php і зазвичай зберігаються в каталозі resources/views.

После встановлення Laravel фреймворка в розпорядженні розробника знаходяться файли app.js і app.css, які представляють собою скомпоновані і мінімізовані jQuery і Bootstrap найостанніших версій на момент виходу релізу Laravel. Якщо стандартних CSS та JS бібліотек не вистачає або ж вони не підходять для розробки, то для використання нових інструментів використовується Laravel Mix. Даний пакет являє собою надбудову над WebPack, що дозволяє розділяти css і js код на окремі модулі, конфігурувати їх використання, налаштовувати мініфікацію і використання css-препроцесорів (sass, less, stylus і т.д.).

З коробки Laravel надає механізм реєстрації і авторизації користувачів, що спрощує життя розробникам, дозволяючи не винаходити чергові велосипеди. Разом з цим, розробники фреймворку добавили валідатори. Laravel валідатори - це конструкції, що дозволяють проводити перевірку даних на підставі різних готових правил. Також Laravel дозволяє створювати власні правила, повідомлення про помилки і кастомні валідатори в цілому.

У Laravel ширико використовується Eloquent ORM. ORM – це технологія програмування, яка покликана полегшити програмістам роботу з БД шляхом надання методів API для типових операцій (вибірка, додавання, оновлення, видалення і т.д.). Реалізацій ORM існує безліч, але творці Laravel і тут заморочили, вигадавши власну. Величезний функціонал Eloquent ORM дозволяє повністю убезпечити себе від атак типу SQL Injection.

У Laravel з коробки доступні інструменти організації черг процесів (наприклад, для масової відправки email). Ця функція незамінна для HighLoad-проектів, тому що дозволяє розвантажити сервер від постійної роботи.

Laravel надає набір методів для створення і управління завданнями, виконуваними за допомогою планувальника завдань Cron. Наприклад завдання з виконанням кожну годину в проміжку між 7 і 22 годинами дня:

```
$Schedule->command ( 'reminders: send') -> hourly () -> between ( '7:00', '22: 00 ');
```

Зручний дебаггінг коду і тестування реалізується наявністю debug panel, спеціальної функції dd() для виведення даних на екран (аналог PHP-конструкції echo '<pre>'; print_r ($ var); die ();).

Що стосується тестування, то в Laravel механізм написання юніт-тестів з використанням популярного тестіровочного фреймворка PHPUnit доступний з коробки. Крім того, в процесі тестування працездатності ресурсів в Laravel фреймворку є можливості емуляції відвідування сторінок сайту і різних дій (натискання на посилання, кнопки, введення тексту і т.д.) завдяки використанню компонентів Symfony.

Оскільки фреймворк – це просто набір інструментів для ефективного написання коду, то все банальні функції, як в CMS, наприклад, додаються за дві секунди шляхом установки готових модулів, в разі використання фреймворків доводиться кожен раз писати руками. Щоб уникнути цієї рутинної роботи, розробники Laravel ввели можливість розширення функціоналу базового програми за рахунок установки пакетів, які є аналогами модулів для CMS.

Ще одна корисна фіча, без якої неможлива розробка повноцінного HighLoad-ресурсу - кешування. Причому, кешування в Laravel доступно за допомогою різних технологій: Redis, MemCached і т.д або за допомогою відповідних драйверів і пакетів. За замовчуванням доступний драйвер кешування file, завдяки якому закешовану інформація буде зберігатися в файлової системі.

Зручний механізм роутінга – це маніпуляції з URL, доступними на сайті, в Laravel неймовірно прості і зручні. Все, що потрібно зробити, для додавання Laravel routes - це відредагувати файл routes/web.php. Найпростіше додавання нового роута виглядає наступним чином:

```
Route :: get ( '/', function () {
return view ( 'welcome');
});
```

Даний код буде виводити на екран вміст файлу resources/views/ welcome.blade.php при переході в корінь сайту.

Також для routes можливо вказувати методи контролерів і призначені для користувача функції, застосовуючи до них MiddleWare – ще одну фішку Laravel, що представляє собою прошарок між route і дією при його виконанні.

Робота з сесіями у Laravel доступна різними способами. Як відомо, об'єкти сесій зберігаються на сервері, отже, ними можна легко і зручно маніпулювати серверними мовами програмування, а також задавати їм різні сховища. Цією особливістю вирішили скористатися Laravel розробники, запровадивши в фреймворк можливість вибору способу зберігання об'єктів сесій за допомогою різних технологій:

- файлове сховище на сервері - стандартні об'єкти сесій
- cookie
- Memcached
- Redis
- Збереження даних в БД
- Тимчасовий PHP масив

2.3 Розробка дизайну

Розробка дизайну веб-системи є настільки ж важливою, як і проектування бази даних. Розрізняють UI та UX дизайн. Обидва тісно зв’язані між собою. UX дизайн можна розшифрувати як досвід користувача або іншими словами – враження користувача від взаємодії з системою. Дизайнер, опираючись на цілі веб-сайту, створює максимально легкий та зрозумілий інтерфейс. UX залежить від великої кількості компонентів, від архітектури веб-сайту та графічного дизайну до чуйності інтерфейсу. Дизайнер повинен продумати наперед кожен крок користувача і втілити це в архітектуру, щоб взаємодія з сайтом була максимально легкою. Після архітектури продуманої архітектури та логіки сайту,

приступають до UI або ж користувацький інтерфейс. Під UI дизайном розуміють графічний інтерфейс системи, набір кольорів, анімацій, кнопки, фотографії та відео – усі елементи, які зроблять веб-систему привабливою. З розвитку дизайну, все більше входить в тенденцію мінімалістичний стиль. Мета будь-якого рекламного сайту донести простими словами до користувачів складні речі і смусити придбати цю ж складну річ. В купі чітко продумані UI та UX можуть зробити практично будь-який дуже прибутковим. Отже, основні особливостями, якими повинен волідити веб-сайт – це ясність інтерфейсу та мінімалістична привабливість. Цим пунктам відповідає один із найпопулярніших дизайнів – Material. Принципи цього дизайну реалізовані в усіх продуктах Google, зокрема в операційній системі Android. Material вирізняється своєю так званою пунктуальністю, гладкими межами та компактністю. В кольоровій гаммі виокремлюють зазвичай два відтінки та похідні від них для підкреслення ховер-ефектів або ж тіней. Основними кольорами для веб-системи мережі кінотеатрів “Metropolitan” було вибрано чорний та сірий. Обов’язковим атрибутом кожного веб-сайту це компонент навігації, який присутній на кожній сторінці. У випадку створюваної веб-системи, використано компонент-шапка, який буде прикліплений до верху сторінки. У шапці знаходиться логотип, який веде на головну сторінку веб-системи, меню та посилання на сторінку авторизації в адмін панель. На основній сторінці знаходиться список фільмів доступних у прокаті. Відповідно після вибору фільму, користувач переходить на сторінку, де відображені усі доступні сеанси у різних кінотеатрах. Після вибору кінотеатра, користувач потрапляє на сторінку з вибором місця у залі. Після оформлення замовлення, повідомлення про заброньоване місце відправляється на пошту користувачу.

2.4 Розробка бази даних

Моделювання бази є однією з найтяжчих та найважливіших етапів розрибки будь-якої системи. Мета моделі даних полягає в тому, щоб переконатися, що всі об'єкти даних, необхідні для бази даних, повністю і точно представлені. Модель даних також є достатньо деталізованою, щоб її використовували розробники баз даних для використання в якості «плану» для створення фізичної бази даних. Інформація, що міститься в моделі даних, буде використана для визначення реляційних таблиць, первинних і зовнішніх ключів, збережених процедур і тригерів. Погано розроблена база даних вимагатиме більше часу в довгостроковій перспективі. Без ретельного планування ви можете створити базу даних, яка виключає дані, необхідні для створення полів, дає неточні або невідповідні результати.

Концептуальна модель представляє в своїй особі тільки таблці та звязки між ними. У нашій базі даних мережі кінотеатрів “Метрополітан”, присутні такі таблиці: кінотеатри, райони, кіностудії, фільми, жанри, сеанси, білети.

Рис. 2.1. Концептуальна модель

Мережа кінотеатрів підрозумовує під собою кілька кінотеатрів, відповідно потрібне місце розташування на певній території. Також кожен кінотеатр має визначену кількість залів. Фільми мають дуже багато особливостей, жанри так кіностудії винесені в окремі таблиці. Таблиця сеансів отримує один фільм та один

зал, відповідно визначену кількість залами білетів. На логічній моделі відображаються поля, ключі та зв’язки між таблицями.

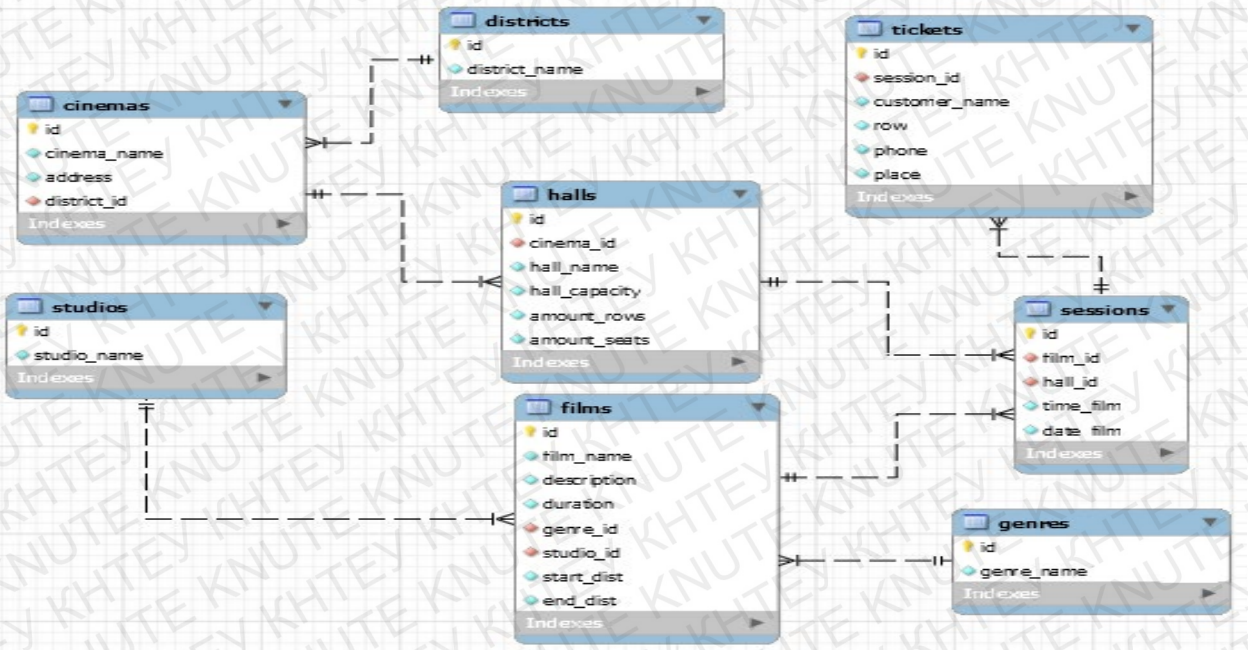

Рис. 2.2. Логічна модель

Відразу потрібно зазначити, що усі таблиці зв’язані між собою за допомогою відносин один до багатьох. Таблиця кінотеатрів має первинне поле id та зовнішній ключ district_id. Таблиця районів має первинне поле id та назву самого району distrcit_name. Кожен кінотеатр має певну кількість залів. Таблиця зали мають первинний ключ, який є і унікальним, також поле до якого кінотеатру відноситься цей зал, назву залу, місткість, кількість залів та сидінь. Варто зазначити, у цій мережі місткість залів стандартизована тому кожен зал має одинакову кількість місць. Таблиця фільмів одна із центральних у цій системі. До неї приєднуються дві інших таблиці жанрів та кіностудій. Таблиця жанри має первинне поле id та поле назви жанра. Таблиця кіностудій має первинне поле id та поле назви кіностудії. Таблиця фільмів має такі поля: первинне поле id, назву фільму, опис, тривалість, зовнішній ключі для жанрів та кіностудій, початок та закінчення прокату. Таблиця сеансів теж одна з центральних таблиць у системі.

Вона має три зв’язки з іншими таблицями. Таблиця сеансів має такі поля: первинне поле id, зовнішні ключі для таблиць фільмів та залів, час та дату сеансу. Також таблиця сеансів має зв’язок з таблицею білетів, відповідно один сеанс має визначену кількістю місць в залі білетів. Таблиця білетів має первинне поле id, зовнішній ключ для зв’язку з таблицею сеансів, назву замовника білета, номер телефону, ряд та місце в залі.Фізична модель представляє собою таблиці зі відносинами та полями. Важливою відмінністю від логічної моделі це те, що у фізичній моделі поля мають тип даних. Усі первинні та зовнішні ключі мають тип даних Integer з довжиною в десять символів. Усі поля які представляють назву якогось об’єкта мають тип даних VARCHAR з довжиною у 255 символів. Поля з датами та часом мають відповідний тип даних. Для дати DATE, для часу Time.

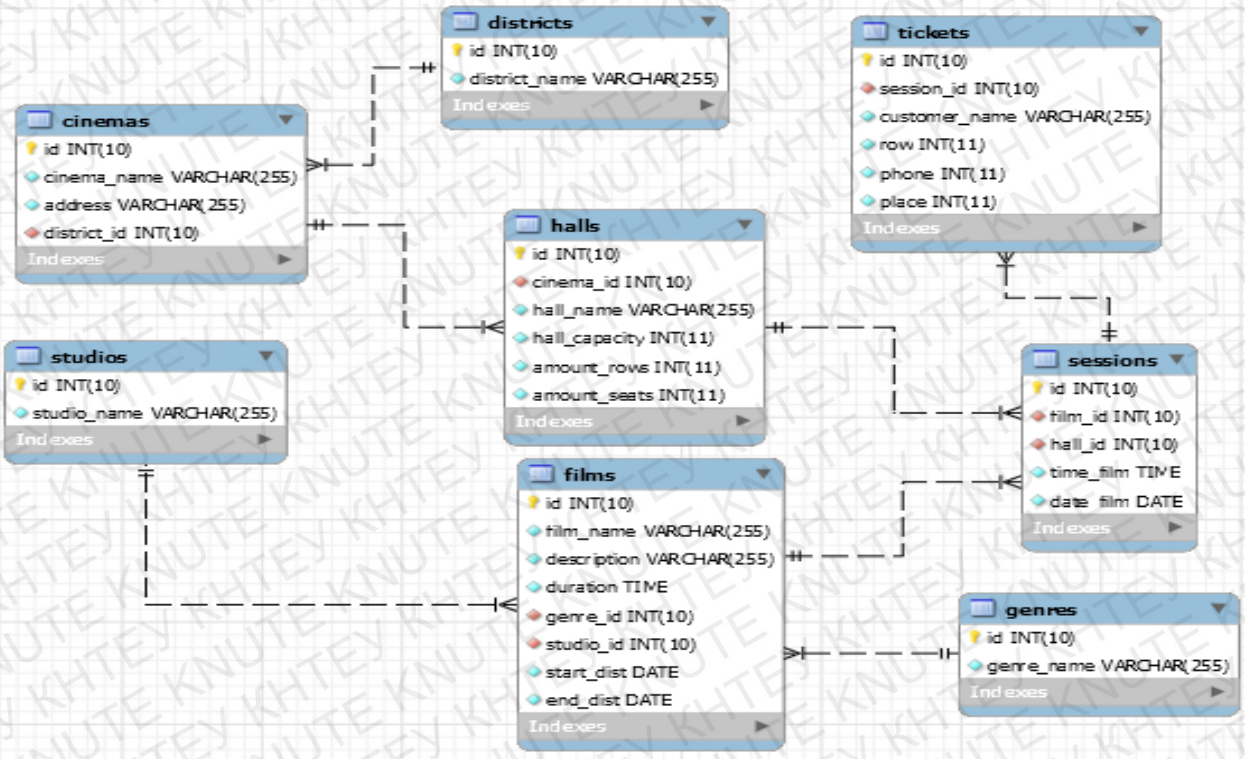

Рис. 2.3 Фізична модель

2.5 Висновки до розділу 2

У другому розділі було розглянуті загальні відомості про програмне забезпечення, фреймворк ларавель. Було проаналізовані особиливості UX та UI

дизайну, за основу вибрано принципи Material підходу та була продумана архітектура і логіка веб-системи. Також були створены концептуальна, логічна та фізичні моделі бази даних.

РОЗДІЛ 3.
СТВОРЕННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 *Реалізація веб-системи*

Для старту розробки будь-якого клієнт серверного забезпечення потрібно налаштувати спеціальне середовище розробки – так званий локальний сервер. Існує достаня кількість інструментів: XAMP, Денвер, Open server. Але найкращим з них є Laragon. Це портативне, ізольоване, швидке і потужне універсальне середовище розробки для PHP, Node.js, Python, Java, Go, Ruby. Laragon пропонує у своєму функціоналі роботу з базами даних з-за допомогою системи адміністрування Heidi SQL та термінал. Отже, Laragon відмінно підходить для побудови та управління сучасними веб-додатками.

Після установки Laragon, з-за допомогою терміналу створюється новий проєкт на laravel:

```
composer global require laravel/installer

laravel new MetropolitanCinema
```

Після створення нового проєкту, першим ділом потрібно налаштувати .env файл. З-за допомогою цього файлу створюється зв’язок з базою даних.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=kurs
DB_USERNAME=root
DB_PASSWORD=
```

Laravel представляє можливість створювати міграції та заповнювати даними за замовчуванням таблиці. Це дуже корисно коли над проектом працюють декілька розробників і усю плутанину з базою даних вирішує одна команда в командній стрічці. Щоб створити міграцію потрібно запустити команду php artisan make:migration ‘назватаблиці‘ –m, додаток –m створить паралельно модель майбуньої таблиці. Моделі потрібні для роботи з базою даних у контроллерах і ще представляють багато інших можливостей.

```
Schema :: create ('cinemas', function (Blueprint $ table) {
        $ table-> increments ('id');
        $ table-> string ('cinema_name');
        $ table-> string ('address');
        $ table-> integer ('district_id') -> unsigned ();
        $ table-> foreign ('district_id') -> references ('id') -> on ('districts');
    });
```

Вище приклад однієї з міграцій, а саме створення таблиці кінотеатрів. Створюється таблиця, поля, первинні та зовнішні ключі і задаються типи даних та окремі універсальні властивості. Таких міграцій усіх 8, як і наших таблиць. Після створення міграцій запускаєм команда php artisan migrate і після цього у нашій базі даних з’являться усі таблиці зі зв’язками.

Після успішного створення усіх таблиць, можна приступати до роботи з кодом. Першим потрібно створити адмін панель для адміністрування даних. Доступ до цієї панелі можуть отримати тільки автентифіковані користувачі, тому потрібно створити автентифікацію. Laravel пропоную вбудовану можливість авторизації для веб-застосунків і для створення її потрібно ввести в термінал наступну команду

```
php artisan ui –auth
```

Якщо операція успішна, то у файловій системі проєкту з’явиться папка auth з налаштованими файлами для авторизації.

Наступним кроком є створення головної сторінки адмін панелі. Усі сторінки зберігаються у папці views. Файли для редагування даних зберігатимуться у папці views/admin. У цій же папці створюється файл admin.blade.php. На цій веб-

сторінці знаходяться посилання на редагування даних таблиць, окрім білетів. Щоб зробити доступною веб-сторінку з-за допомогою браузера потрібно у файлі routes/web.php створити маршрут до файлу admin.blade.php.

```
Route::get('/admin', function () { return view('admin.admin'); });
```

Веб-сторінка буде доступною за посиланням домен/admin.

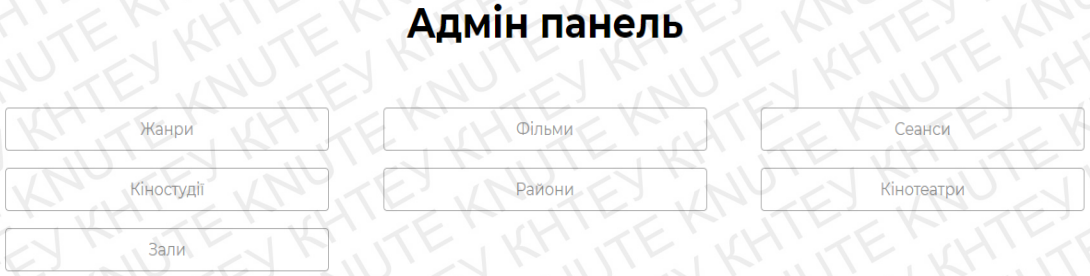

Рис. 3.1. Адмін панель

Для редагування даних у Laravel використовуються контрллери типу resource. Для кожної моделі можна створити декілька контроллерів. Для роботи з адміністраторською панеллю створюється окремі контроллери. У терміналі потрібно ввести команди:

```
php artisan make:Controller Admin/DistrictController –r
php artisan make:Controller Admin/CinemaController –r
php artisan make:Controller Admin/FilmController –r
php artisan make:Controller Admin/GenreController –r
php artisan make:Controller Admin/StudioController –r
php artisan make:Controller Admin/HallController –r
php artisan make:Controller Admin/SessionsController –r
```

Усі контроллери зберігаються у папці app/http/controllers. У нашому випадку приставка Admin/ означає, що контроллер буде створений у app/http/controllers/Admin. Ключ –r означає, що контроллер повинен бути створений типу resource. Для роботи з базою даних використовуються об’єкт моделі контретної таблиці або глобальний об’єкт бази даних, які потрібно додатково підключають.

```
use App\Models\Cinema;
```

```
use Illuminate\Support\Facades\DB;
```

Ресурсний контроллер пропонує 7 методів: index, show($id), create, edit($id), store, destroy($id), update($id), де $id – це змінна, в яку передається id запису з таблиці даних. Щоб отримати доступ до цих методів використовуються файл routes/web.php. У цьому файлі прописуються маршрути до цих методів.

```
//// жанри
$groupData = [
    'namespace' => 'Admin',
    'prefix'    => 'admin',
];
Route::group($groupData , function () {
    //genres
    Route::resource('genres','GenresController')->names('admin.genres');
    Route::resource('studios','StudioController')->names('admin.studios');
    Route::resource('districts','DistrictController')->names('admin.districts');
    Route::resource('cinemas','CinemaController')->names('admin.cinemas');
    Route::resource('halls','HallController')->names('admin.halls');
    Route::resource('films','FilmController')->names('admin.films');
    Route::resource('sessions','SessionController')->names('admin.sessions');
});
```

Методи у шаблонах blade.php доступні з-за допомогою рядка {{ route('admin.cinemas.назва методу) }}.

Зазвичай, метод index використовується показу усіх даних в таблиці.

```
public function index() {
        $districts = DB::table ('districts')->get();
        $items = DB::table ('cinemas')
                -> join ('districts', 'cinemas.district_id', '=', 'districts.id')
                -> select ('cinemas.*', 'districts.district_name')
                -> get ();
        return view('admin.cinemas', compact('items','districts'));
}
```

В даному випадку дані отримуються з-за допомогою глобального об’єкту DB. У змінній $districts зберігаються усі райони, а у змінній $items – кінотеатри. Рядок return view(‘admin.cinemas’, compact(‘items’,’districts’)) повертає маршрут на шаблон редагування кінотеатрів та передає в цей шаблон отримані з бази даних дані.

```
@extends('layouts.app')
@section('content')
<div class="container">
  <a href="{{ url('/admin') }}" class="back2admin"><i class="fas fa-long-arrow-alt-left"></i>Адмін панель</a>
</div>
<h2>Кінотеатри</h2>
<div class="container">
  <div class="add">
    <a class="add_head">Добавити <i class="fas fa-plus"></i></a>
    <div class="add_body">
      <hr>
      <form action="{{ route('admin.cinemas.store') }}" method="POST">
        @csrf
        <div class="group_input">
          <label for="cinema_name">Назва кінотеатру</label>
          <input type="text" name="cinema_name" placeholder="Введіть назву кінотеатру" id="cinema_name">
        </div>
        <div class="group_input">
          <label for="cinema_address">Адреса кінотеатру</label>
          <input type="text" name="cinema_address" placeholder="Введіть адресу кінотеатру" id="cinema_address">
        </div>
        <div class="group-select">
          <label for="district_id">Район</label>
          <select name="district_id">
```

```
                <option value="-1"></option>
                @foreach( $districts as $district)
                <option value="{{ $district->id }}">{{ $district->district_name }}</option>
                @endforeach
              </select>
            </div>
            <button class="submit" type="submit">Зберігти</button>
          </form>
        </div>
      </div>
</div>
<div class="wrapper_table">
    <div class="container">
      <table cellspacing="0">
        <thead>
          <tr>
            <th>№</th>
            <th>Назва жанру</th>
            <th>Назва району</th>
            <th>Адреса</th>
            <th></th>
          </tr>
        </thead>
        @foreach( $items as $key => $item)
        <tr>
          <td>{{ $key+1 }}</td>
          <td><a  href="{{  route('admin.cinemas.edit',  $item->id)  }}">{{  $item->cinema_name
}}</a></td>
          <td>{{ $item->district_name }}</td>
          <td>{{ $item->cinema_address }}</td>
          <td>
            <form action="{{ route('admin.cinemas.destroy' , $item->id ) }}" method="POST">
              {{ csrf_field() }}
```

```
                {{ method_field('DELETE') }}
                <button type="submit"><i class="far fa-trash-alt"></i></button>
            </form>
        </td>
    </tr>
    @endforeach
  </table>
 </div>
</div>
@endsection('content')
```

В Laravel можна використати глобальні шаблони. Тому для створення шапки сайту, яка знаходиться на кожній сторінці використовується саме цей метод. Він реалізований з-за допомогою рядку @extends('layouts.app'). Увесь код, який знаходиться між рядками @section(‘content’) і @endsection(‘content’) реєструється в глобальну секцію content і динамічно імпортується у файл views/layouts/app.blade.php.

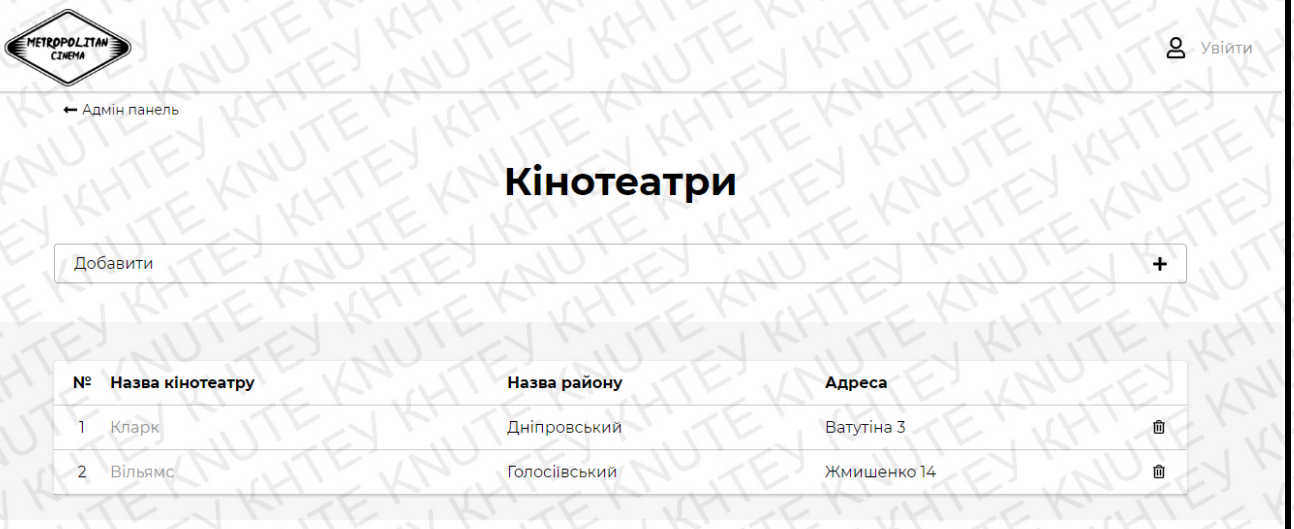

Рис. 3.2. Шаблон редагування даних

На даному шаблоні відображені уже добавлені кінотеатри та присутня форма добавлення кінотеатру. За створення кінотеатру відповідає метод store.

```
$data = $request->input();
$item = new Cinema($data);
$result = $item->save();
```

```
        if($result){
            return redirect()->route('admin.cinemas.index');
        }
```

З-за допомогою рядку $data = $request->input() отримуються дані з полів форми. Варто зазначити, що назви полів форми повинні співпадати з назвами полів у таблиці. $item = new Cinema($data) – створюється новий об’єкт моделі Cinema, а з допомогою методу save(), дані зберігаються у базі. Якщо операція успішна, то адміністратора редіректить на маршрут (‘admin.cinemas.index’).

Метод store викликається у формі з-за допомогою маршрута - <form action="{{ route('admin.cinemas.store') }}" method="POST">. Важливо зазначити, що без @crsf форма працювати не буде, це невід’ємний елемент в кіберзахисті веб-системи.

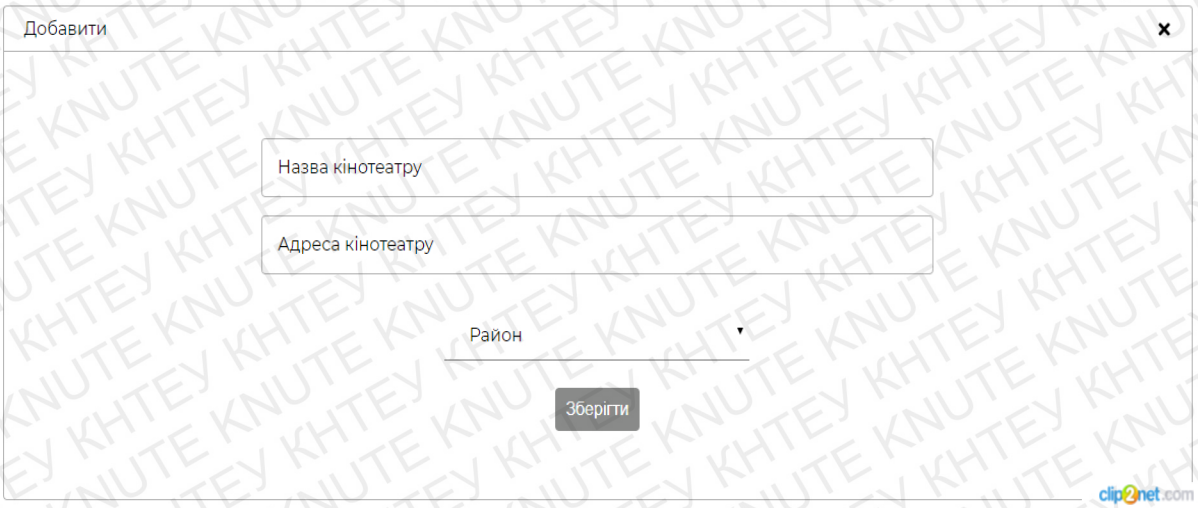

Рис. 3.3. Форма додавання кінотеатру

Видалення запису з бази даних реалізується з-за допомогою методу destroy.

```
public function destroy($id)
    {
        $item = Cinema::find($id);
        $result = $item->delete();
        if($result){
            return redirect()->route('admin.cinemas.index');
        }
```

```
    }
```

У функцію передається id запису і з-за допомогою вбудованого метода delete() об’єкт успішно видаляється з бази даних. Після видалення користувача перенаправляє на маршрут admin.cinemas.index. Редагування запису відбувається з-за допомогою методу edit. У функцію передається id запису, отримується уся необхідна інформація для редагування запису і користувача перенаправляє на нову сторінку, де і відбувається операція.

```
public function edit($id)
{
    $districts = DB::table('districts')->get();
    $item = DB::table ('cinemas')->where('cinemas.id','=',$id)
        -> join ('districts', 'cinemas.district_id', '=', 'districts.id')
        -> select ('cinemas.*', 'districts.*')
        -> first ();
    return view('admin.cinemas.edit', compact('item','districts'));
}
```

Для збереження зміненої інформації використовується метод update.

```
public function update(Request $request, $id)
{
    $item = Cinema::find($id);
    if(empty($item)){
        return back()->withErrors(['msg'=>'Район не знайдений']);
    }
    $data = $request->all();
    $result = $item->fill($data)->save();
    if($result){
        return redirect()->route('admin.cinemas.index');
    }
}
```

Якщо інформація успішно збережеться, то користувача перенаправить на основну сторінку редагування кінотеатрів.

На головній сторінці веб-системи користувача зустрічає список доступних у прокаті фільмів. За відображення цієї сторінки відповідає контроллер MainController, а саме метод index.

```
public function index()
{
    $today = date("Y/m/d");
    $items = DB::table('films')->where('end_dist', '>=', $today)->get();
    return view('welcome', compact('items'));
}
```

З-за допомогою цього методу, отримуються фільми, де дата закінчення прокату сьогоднішній або день у майбутньому.

```
<div class="container">
  <div class="films">
    @foreach ($items as $item)
    <div class="film">
      <div class="figure">
        <img src="{{url('storage/'.$item->avatar)}}">
      </div>
      <div class="film-title">
        <h2>{{ $item->film_name}}</h2>
      </div>
      <a href="{{ route('film', $item->id) }}">Дивитись</a>
    </div>
    @endforeach
  </div>
</div>
```

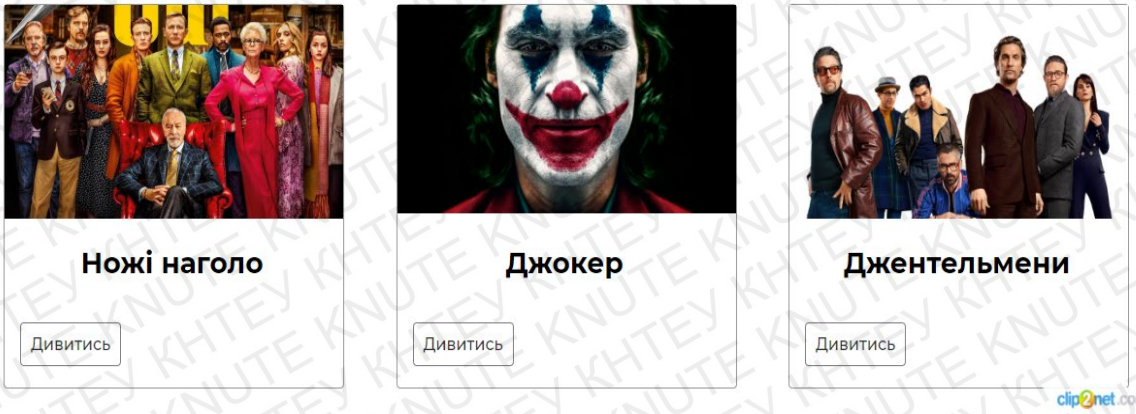

Рис. 3.4. Головна сторінка

Щоб перейти на сторінку фільму, потрібно натиснути на кнопку “Дивитись”. Опис, жанр, кіностудія та тривалість фільму винесена в правий блок. Сеанси винесені в окремий блок під обкладинкою фільму. В списках сеансів відображається кінотеатр в якому відбудеться показ фільму, дата та час початку.

Рис. 3.5. Сторінка фільму

Після вибору сеансу, користувач переходить на сторінку вибору місця у залі. Чорним кольором залиті комірки, які уже заброньовані.

```
<span class="rows">{{ $item->amount_rows}}</span>
<span class="seats">{{ $item->amount_seats}}</span>
```

```
<div class="session_single_seats">
  <div class="places">
  </div>
</div>
```

З-за допомогою Jquery, отримуються значення з тегів . На основі отриманих значень кількості рядів та кількості місць у ряді будується зал, де комірка – це місце.

```
var rows = Number.parseInt($('.rows').text());
var seats = Number.parseInt($('.seats').text());
for(let i = 1; i < rows; i++){
  $('.places').append('<div class="row" id="row'+ i +'"></div>')
  for(let y = 1; y < seats+1; y++){
    $('#row'+i).append('<span class="seat" id="place'+ y +'">'+ y +'</span>')
  }
}
```

Рис. 3.6. Сторінка бронювання білету

Для вибору місця потрібно натиснути на комірку.

```
$('body').delegate('.seat','click', function(){
  if($(this).hasClass('bron') !== true){
    $('.seat').removeClass('active_seat');
    $(this).toggleClass('active_seat');
    $('.form_seat').val($(this).text());
    $('.form_row').val($(this).parent().prop('id').substr(3));
  }
});
```

Після натиску на комірку, номер ряду та номеря місця динамічно з-за допомогою Jquery переноситься в інпути типу hidden. Таким чином у формі знаходиться уся необхідна інформація для збереження бронювання місця.

```
<form action="{{ route('session.store') }}" method="POST">
  @csrf
  <input type="hidden" name="session_id" value="{{$item->id}}">
  <input type="hidden" name="row" class="form_row" value="">
  <input type="hidden" name="place" class="form_seat" value="">
  <div class="group_input">
    <label for="customer_name">Ім'я</label>
    <input type="text" name="customer_name" placeholder="Введіть ваше ім'я"
id="customer_name">
  </div>
  <div class="group_input">
    <label for="phone">Номер телефону</label>
    <input type="text" name="phone" placeholder="Введіть ваш номер телефону"
id="phone">
  </div>
  <div class="group_input">
    <label for="email">E-mail</label>
    <input type="email" name="email" placeholder="Введіть ваш e-mail"
id="email">
  </div>
  <button class="submit" type="submit">Зберігти</button>
</form>
```

Уся інформація зберігається з-за допомогою метода store контроллера SessionController. Також інформація про забронюване місце, а саме фільм, зал, місце, ряд, дата і час відправляються користувачеві на e-mail. Після успішного бронювання клієнта перенаправляє на головну сторінку.

3.2 Висновок до розділу 3

Отже, на основі створеної бази даних та дизайну Material було створено веб-системи мережі кінотеатрів “Метрополітан”. Під створення веб-застосунку використовувались такі інструменти: ларавель, Javascript та бібліотека Jquery, HTML та CSS. Інтерфейс для користувача та адміністратора не викличе ніяких труднощів у використанні.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В даній випускній кваліфікаційній роботі була проведена широкомастабна робота. Був пройдений повний шлях від продумання дизайну до створення готової веб-системи.

Проаналізувавши способи відпочинку людей за тисячолітню історію, зрозуміло що кіно є одним з найпопулярніших способів розваг. Кіноіндустрія розрослась до невимовно великих розмірів. Кожного року випускається тисячі фільмів і заробляються мільйони доларів. Саме тому для обширного показу та отримання прибку від знятих фільмів були створені кінотеатр, як спосіб розповсюдження фільмів. В сучасному світі уже давно появилась можливість бронювати білети в кінотеатри, а для опарцювання такої великої кількості інформації потрібні бази даних.

На основі загального аналізу web-систем було створено технічне завдання. Основними тезисами, щодо розробки є лаконічність дизайну та швидкість веб-системи. За основний інструмент розробки було вибрано Laravel. Цей фреймворк сильно виділяється на фоні інших PHP-інструментів,що і підкреслює його популярність у IT середовищі. Для досягнення лаконічного дизайну, було використано підхід Material, розроблений компанією Google. Висока популярність протягом довгих років доводить актуальність цих принципів. Для створення бази даних було використано систему управління базами даних Mysql, а також систему адміністрування СУБД HeidiSQL. Під час розробки моделі бази даних було враховано, що “Метрополітан” – це мережа кінотеатрі, а тому існують декілька кінопрокатних місць у певних районах. Усі моделі було створені успішно і прекрасно підійшли для проекту, задовільнивши усі потреби для опрацювання даних.

Програмний продукт створений у вигляді веб-застосунку з використанням Laragon як середовища розробки. Веб-сайт виконаний у мінімалістичному стилі і з легким у плані навігації дизайні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

Основний

1. Бен Форта. SQL за 10 минут / Б.Форта: Діалектика, 2020. – 288 с.
2. Лінн Байлі. Head First SQL: Your Brain on SQL – A Learner's Guide / Лінн Байлі: O-Reilly Media, 2007. – 592 с.
3. Адам Трагтенрберг. PHP CookBook PHP Cookbook: Solutions & Examples for PHP Programmers 3rd Edition / А. Трагтенрберг, Д.Склар: O-Reilly Media , 2014. – 840 с.
4. Девід Фленган. Javascript: The definitive Guide / Д.Флеган: O-Reilly Media, 2011. – 1100 с.
5. Джон Дакетт. Javascript и jQuery. Интерактивная разработка / Д.Дакетт: Эксмо, 2017. – 640 с.
6. Дженніфер Роббінс. HTML 5. Карманный справочник / Д.Роббінс: Діалектика, 2020. – 192 с.
7. Естель Вейль. Характеристики CSS. Полный справочник. Визуальное форматирование веб-страниц / Е.Вейль, Е.Майер: Вільямс, 2016. – 1088 с.

Інтернет ресурси

1. Введення до SQL [Електронний ресурс] – Режим доступу до ресурсу: https://launchschool.com/books/sql/ (дата звернення 01.05.2020).
2. Laravel framework for PHP [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/7.x (дата звернення 01.05.2020).
3. W3schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/ (дата звернення 01.05.2020).

ДОДАТКИ

Додаток А

```
composer global require laravel/installer

laravel new MetropolitanCinema

        DB_CONNECTION=mysql
        DB_HOST=127.0.0.1
        DB_PORT=3306
        DB_DATABASE=kurs
        DB_USERNAME=root
        DB_PASSWORD=
        Schema :: create ('cinemas', function (Blueprint $ table) {
                $ table-> increments ('id');
                $ table-> string ('cinema_name');
                $ table-> string ('address');
                $ table-> integer ('district_id') -> unsigned ();
                $ table-> foreign ('district_id') -> references ('id') -> on ('districts');
            });
php artisan ui –auth
Route::get('/admin', function () { return view('admin.admin'); });
        php artisan make:Controller Admin/DistrictController –r
        php artisan make:Controller Admin/CinemaController –r
        php artisan make:Controller Admin/FilmController –r
        php artisan make:Controller Admin/GenreController –r
        php artisan make:Controller Admin/StudioController –r
        php artisan make:Controller Admin/HallController –r
        php artisan make:Controller Admin/SessionsController –r
        use App\Models\Cinema;
        use Illuminate\Support\Facades\DB;
//// жанри
$groupData = [
   'namespace' => 'Admin',
   'prefix'    => 'admin',
];
```

```
Route::group($groupData , function () {
  //genres
  Route::resource('genres','GenresController')->names('admin.genres');
  Route::resource('studios','StudioController')->names('admin.studios');
  Route::resource('districts','DistrictController')->names('admin.districts');
  Route::resource('cinemas','CinemaController')->names('admin.cinemas');
  Route::resource('halls','HallController')->names('admin.halls');
  Route::resource('films','FilmController')->names('admin.films');
  Route::resource('sessions','SessionController')->names('admin.sessions');
});
public function index() {
        $districts = DB::table ('districts')->get();
        $items = DB::table ('cinemas')
                -> join ('districts', 'cinemas.district_id', '=', 'districts.id')
                -> select ('cinemas.*', 'districts.district_name')
                -> get ();
        return view('admin.cinemas', compact('items','districts'));
}
@extends('layouts.app')
@section('content')
<div class="container">
  <a href="{{ url('/admin') }}" class="back2admin"><i class="fas fa-long-arrow-alt-left"></i>Адмін
панель</a>
</div>
<h2>Кінотеатри</h2>
<div class="container">
  <div class="add">
    <a class="add_head">Добавити <i class="fas fa-plus"></i></a>
    <div class="add_body">
      <hr>
      <form action="{{ route('admin.cinemas.store') }}" method="POST">
        @csrf
        <div class="group_input">
          <label for="cinema_name">Назва кінотеатру</label>
```

```
            <input type="text" name="cinema_name" placeholder="Введіть назву кінотеатру" id="cinema_name">
          </div>
          <div class="group_input">
            <label for="cinema_address">Адреса кінотеатру</label>
            <input type="text" name="cinema_address" placeholder="Введіть адресу кінотеатру" id="cinema_address">
          </div>
          <div class="group-select">
            <label for="district_id">Район</label>
            <select name="district_id">
              <option value="-1"></option>
              @foreach( $districts as $district)
              <option value="{{ $district->id }}">{{ $district->district_name }}</option>
              @endforeach
            </select>
          </div>
          <button class="submit" type="submit">Зберігти</button>
        </form>
      </div>
    </div>
  </div>
  <div class="wrapper_table">
    <div class="container">
      <table cellspacing="0">
        <thead>
          <tr>
            <th>№</th>
            <th>Назва жанру</th>
            <th>Назва району</th>
            <th>Адреса</th>
            <th></th>
          </tr>
```

```
        </thead>
        @foreach( $items as $key => $item)
        <tr>
          <td>{{ $key+1 }}</td>
          <td><a href="{{ route('admin.cinemas.edit', $item->id) }}">{{ $item->cinema_name
}}</a></td>
          <td>{{ $item->district_name }}</td>
          <td>{{ $item->cinema_address }}</td>
          <td>
            <form action="{{ route('admin.cinemas.destroy' , $item->id ) }}" method="POST">
              {{ csrf_field() }}
              {{ method_field('DELETE') }}
              <button type="submit"><i class="far fa-trash-alt"></i></button>
            </form>
          </td>
        </tr>
        @endforeach
      </table>
    </div>
</div>
@endsection('content')
        $data = $request->input();
            $item = new Cinema($data);
            $result = $item->save();
            if($result){
              return redirect()->route('admin.cinemas.index');
            }
        public function destroy($id)
          {
            $item = Cinema::find($id);
            $result = $item->delete();
            if($result) {
```

```
            return redirect()->route('admin.cinemas.index');
        }
    }
    public function edit($id)
    {
        $districts = DB::table('districts')->get();
        $item = DB::table ('cinemas')->where('cinemas.id','=',$id)
            -> join ('districts', 'cinemas.district_id', '=', 'districts.id')
            -> select ('cinemas.*', 'districts.*')
            -> first ();
        return view('admin.cinemas.edit', compact('item','districts'));
    }
     public function update(Request $request, $id)
    {
        $item = Cinema::find($id);
        if(empty($item)){
            return back()->withErrors(['msg'=>'Район не знайдений']);
        }
        $data = $request->all();
        $result = $item->fill($data)->save();
        if($result){
            return redirect()->route('admin.cinemas.index');
        }
    }
    public function index()
    {
        $today = date("Y/m/d");
        $items = DB::table('films')->where('end_dist', '>=', $today)->get();
        return view('welcome', compact('items'));
    }
<div class="container">
```

```
    <div class="films">
      @foreach ($items as $item)
      <div class="film">
        <div class="figure">
          <img src="{{url('storage/'.$item->avatar)}}">
        </div>
        <div class="film-title">
          <h2>{{ $item->film_name}}</h2>
        </div>
        <a href="{{ route('film', $item->id) }}">Дивитись</a>
      </div>
      @endforeach
    </div>
</div>
      <span class="rows">{{ $item->amount_rows}}</span>
      <span class="seats">{{ $item->amount_seats}}</span>
      <div class="session_single_seats">
        <div class="places">
        </div>
      </div>
var rows = Number.parseInt($('.rows').text());
    var seats = Number.parseInt($('.seats').text());
    for(let i = 1; i < rows; i++){
      $('.places').append('<div class="row" id="row'+ i +'"></div>')
      for(let y = 1; y < seats+1; y++){
        $('#row'+i).append('<span class="seat" id="place'+ y +'">'+ y +'</span>')
      }
    }
    $('body').delegate('.seat','click', function(){
      if($(this).hasClass('bron') !== true) {
```

```
            $('.seat').removeClass('active_seat');
            $(this).toggleClass('active_seat');
            $('.form_seat').val($(this).text());
            $('.form_row').val($(this).parent().prop('id').substr(3));
        }
    });
<form action="{{ route('session.store') }}" method="POST">
            @csrf
            <input type="hidden" name="session_id" value="{{$item->id}}">
            <input type="hidden" name="row" class="form_row" value="">
            <input type="hidden" name="place" class="form_seat" value="">
            <div class="group_input">
              <label for="customer_name">Ім'я</label>
              <input type="text" name="customer_name" placeholder="Введіть ваше ім'я"
id="customer_name">
            </div>
            <div class="group_input">
              <label for="phone">Номер телефону</label>
              <input type="text" name="phone" placeholder="Введіть ваш номер телефону"
id="phone">
            </div>
            <div class="group_input">
              <label for="email">E-mail</label>
              <input type="email" name="email" placeholder="Введіть ваш e-mail"
id="email">
            </div>
            <button class="submit" type="submit">Зберігти</button>
</for>
```