**ЛИСТ ЕЖЕДНЕВНОЙ РАБОТЫ СТУДЕНТА**

|  |  |  |
| --- | --- | --- |
| Дата  (по графику практики) | Выполненная работа студента | Оценка руководителя практики  (за каждый день) |
|  | *Инструктажи по охране труда, технике безопасности, пожарной безопасности и ознакомлению с Правилами внутреннего трудового распорядка организации* |  |
| *5.05.2025* | Сегодня началась моя учебная практика по теме «Кинотеатр». В первую очередь прошёл инструктажи по охране труда, технике безопасности, пожарной безопасности, а также ознакомился с правилами внутреннего трудового распорядка учебного учреждения. Эти мероприятия являются неотъемлемой частью начала любой профессиональной деятельности и крайне важны для обеспечения безопасных условий труда и взаимодействия с электронными системами, офисным оборудованием и коллегами в рамках проекта.На инструктаже были рассмотрены такие темы, как правила поведения в случае возникновения пожара, действия при эвакуации, порядок использования первичных средств пожаротушения (огнетушителей), а также меры по предотвращению короткого замыкания при работе с электрическими приборами. Кроме того, особое внимание было уделено технике безопасности при работе за компьютером: организация рабочего места, соблюдение режима труда и отдыха, предотвращение переутомления глаз и опорно-двигательной системы. Также была дана информация о недопустимости использования неисправного оборудования и необходимости сообщать о любых сбоях в работе ответственным лицам.Далее был организован инструктаж по внутреннему распорядку организации: изучил режим работы, правила посещаемости, порядок подачи отчётных материалов и требования к дисциплине. Руководитель практики ознакомил меня с проектной документацией и структурой технического задания, которое мне предстоит разработать и реализовать в течение практики.Мной была изучена тема проекта: «Кинотеатр» — веб-приложение, которое будет состоять из пользовательской части и административной панели. Основные функции включают регистрацию и авторизацию пользователей, просмотр списка фильмов, выбор сеансов, бронирование мест, оформление заказов, а также отображение истории бронирований и отзывов.  День завершился подготовкой черновика документа с описанием требований и составлением архитектурной. Также я провёл настройку рабочего окружения на своём компьютере: установил необходимое ПО Node.js, npm, PHP, Composer, MySQL, Laravel |  |
| *6.05.2025* | На втором дне учебной практики я приступил к первому этапу проектирования — разработке технического задания (ТЗ) на веб-приложение «Кинотеатр». Этот этап является фундаментом для всей дальнейшей работы, поскольку от полноты и точности составленного ТЗ зависит эффективность реализации проекта, соответствие готового продукта ожиданиям пользователей и возможность последующего масштабирования системы.Сначала я провёл анализ требований к функционалу, который должен быть реализован в веб-приложении. Основные функции включают просмотр списка фильмов, подробную страницу каждого фильма, выбор даты и времени сеанса, бронирование мест в зале, авторизацию и регистрацию пользователей, управление личным кабинетом, просмотр истории бронирований и отзывов, а также административную панель для управления контентом (фильмами, сеансами, пользователями, бронированиями и статистикой). Отдельно были выделены роли пользователей: обычный пользователь и администратор.Я подробно расписал каждую страницу приложения. Например, главная страница должна отображать афишу фильмов — с изображениями, названиями, жанрами и рейтингом. При нажатии на карточку фильма пользователь переходит на страницу с описанием, трейлером и кнопкой «Забронировать билет». На странице бронирования отображается план зала с возможностью выбора свободных мест, ввод личных данных и подтверждение заказа. Также предусмотрена форма обратной связи и возможность оставить отзыв после посещения киносеанса.Далее я определил архитектуру взаимодействия между фронтендом и бэкендом. Были описаны все необходимые REST API-эндпоинты, их методы (GET, POST, PUT, DELETE), входные параметры и возвращаемые данные. Например, для получения списка фильмов нужен маршрут GET /api/movies, а для создания бронирования — POST /api/bookings. Также предусмотрена система авторизации через JWT-токены, чтобы обеспечить безопасность пользовательских данных и разграничение доступа.Помимо этого, я составил схему базы данных. В ней предусмотрены таблицы: users, movies, sessions, bookings, reviews, seats. Таблица users хранит данные о пользователях: имя, email, пароль, роль. В таблице movies содержится информация о фильмах: название и тд. По итогам дня я составил документ в формате .docx с полным текстом технического задания, таблицей API, схемой базы данных и скриншотами архитектурной схемы. Таким образом, второй день практики завершился завершением важнейшего этапа — формализацией требований, на основании которых в последующие дни. |  |
| *7.05.2025* | Сегодня я продолжил работу над проектом «Кинотеатр» и завершил формирование технического задания, а также начал реализацию backend-части веб-приложения с использованием фреймворка Laravel. Этот день стал переходным этапом от теоретического проектирования к непосредственной разработке. Главной задачей было перевести ранее подготовленные архитектурные решения и структуру базы данных в реальные миграции, модели и контроллеры в Laravel-приложении.В начале дня я создал новый Laravel-проект с помощью команды laravel new cinema-backend, настроил файл .env для подключения к базе данных MySQL, проверил соединение и инициализировал репозиторий Git для отслеживания изменений. После успешной настройки окружения приступил к созданию миграций и моделей для таблиц users, movies, sessions, bookings, reviews и seats. Каждая миграция включала необходимые поля, типы данных и связи. Например, таблица sessions содержит внешний ключ movie\_id, связанный с таблицей movies, а таблица bookings — ссылки на пользователя, сеанс и занятые места.Далее я реализовал модели Eloquent, описал связи между ними: User имеет множество Bookings, Movie имеет много Sessions, Session имеет много Bookings. Это обеспечит простоту взаимодействия с данными и сократит дублирование кода при построении логики приложения. Также были добавлены кастомные методы, например availableSeats() для модели Session, чтобы легко получать количество доступных мест на конкретный сеанс.Параллельно с моделями и миграциями я создал базовые контроллеры: AuthController, MovieController, SessionController, BookingController, ReviewController. Для начала реализовал следующие методы: регистрация (register), вход (login), получение списка фильмов (index), создание сеанса, просмотр доступных сеансов на определённый фильм, создание бронирования, получение истории бронирований пользователя. Настроил роутинг в routes/api.php и протестировал работу маршрутов через Postman. Для авторизации использовал Laravel Sanctum с токенами. Настроил middleware, ограничивающий доступ к защищённым маршрутам.Особое внимание уделил валидации входных данных через FormRequest и правильной обработке ошибок. Также внедрил API-ответы в едином формате с полем status, message и data. Это поможет frontend-разработке быстрее обрабатывать ответы от сервера.  Также я добавил seeder-файлы для генерации тестовых данных. В таблицу movies было добавлено 3 фильмов с разными жанрами. Завершил день фиксацией текущего состояния проекта |  |
| *8.05.2025* | Сегодняшний день был посвящён углублённой разработке backend-логики веб-приложения «Кинотеатр» на Laravel. Основной задачей было продолжить реализацию REST API, охватывая как пользовательские, так и административные функции. Я работал над расширением функционала контроллеров, добавлением новых маршрутов, тестированием запросов и доработкой базы данных. Особое внимание уделялся логике работы с сеансами, местами в зале и бронированиями.Начал с улучшения функционала модели Session, чтобы отображать доступные места для каждого сеанса. Я добавил метод, который рассчитывает количество свободных и занятых мест, учитывая уже оформленные бронирования. В таблице seats теперь хранятся сведения о статусе каждого места (свободно/занято) и его принадлежности к определённому сеансу. Это позволило обеспечить точную и наглядную информацию для клиента, когда пользователь выбирает места на странице бронирования.Далее расширил функциональность BookingController. Теперь пользователь может не просто создать бронирование, но и получить список всех своих заказов с фильтрацией по дате, сеансу или фильму. Также добавил функциональность отмены бронирования (удаление записи с возвратом мест в статус «свободно»). Это потребовало реализации механизма транзакций в Laravel, чтобы обеспечить целостность данных при одновременной записи в несколько таблиц (bookings, seats, sessions).Параллельно реализовал ReviewController, чтобы пользователи могли оставлять отзывы на просмотренные фильмы. Каждый отзыв содержит текст, рейтинг (от 1 до 5 звёзд), а также ID фильма и пользователя. Настроил ограничение: отзыв можно оставить только один раз после посещения сеанса. Для этого валидация проверяет наличие действительного бронирования на прошедший сеанс. Это защищает систему от фейковых отзывов и делает систему более реалистичной.В административной части реализовал маршруты и методы для создания, редактирования и удаления фильмов и сеансов. Добавил защиту маршрутов с помощью middleware is\_admin. При попытке доступа обычного пользователя к административным ресурсам возвращается ошибка 403. Это позволяет разграничить права доступа и обеспечить безопасную работу с данными. Также добавлены кастомные политики (policies) для каждой сущности. В завершение дня провёл «рефакторинг» кода  Под конец рабочего дня зафиксировал все изменения в Git с соответствующими коммитами и подробно описал сделанное в отчётном файле. |  |
| *10.05.2025* | Пятый день учебной практики был полностью посвящён тестированию и завершению базового функционала backend-приложения, а также переходу к подготовке frontend-части на ReactJS. Я сфокусировался на устранении мелких недоработок, улучшении логики безопасности и оптимизации API, а во второй половине дня начал реализацию клиентской части приложения, которая будет взаимодействовать с уже готовыми backend-эндпоинтами.Первым делом я провёл серию интеграционных тестов. Проверил корректность авторизации и регистрации пользователей: защита от дублирующих email, валидация ввода, корректность создания токена и его отправка в заголовке. Далее протестировал все маршруты по работе с фильмами и сеансами: получение списка фильмов, подробностей по каждому, доступных сеансов, создание и отмену бронирований. Проверялась реакция сервера на граничные случаи — попытка забронировать место на прошедший сеанс, бронирование одного и того же места двумя пользователями, попытка доступа к административной панели без прав администратора и т.д.После устранения всех найденных ошибок я добавил улучшенную систему логирования ошибок, включающую сохранение запроса, IP-адреса, ID пользователя и текста ошибки в отдельный лог-файл. Это поможет при анализе проблем в будущем. Также реализовал простую систему ограничений на количество запросов (rate limiting) с использованием Laravel middleware, чтобы предотвратить спам и чрезмерную нагрузку на сервер.Далее перешёл к созданию frontend-проекта. Использовал Vite для быстрой и лёгкой инициализации React-приложения. Командой npm create vite@latest cinema-frontend --template react был создан проект, после чего я установил базовые зависимости: react-router-dom для маршрутизации, axios для отправки HTTP-запросов, antd — библиотеку пользовательских компонентов для UI. Настроил структуру проекта с разделением на pages, components, services и utils.Создал главный layout приложения с навигационной панелью: ссылки на страницы фильмов, историю заказов, вход и регистрацию. Использовал Ant Design компоненты Layout, Menu, Button и другие. Добавил начальную маршрутизацию с помощью BrowserRouter, Routes и Route, реализовал базовую навигацию между страницами. Добавлена защита маршрутов с помощью проверки наличия токена и состояния авторизации в приложении. Впереди — реализация страницы бронирования, отображение сеансов и логика выбора мест. Завершил день фиксацией всех изменений, коммитом в Git и созданием краткого документа с описанием реализованных страниц и компонентов. |  |
| *12.05.2025* | Шестой день практики стал важной вехой в создании клиентской части проекта «Кинотеатр». Я сосредоточился на реализации ключевого функционала — отображения подробной информации о фильмах, сеансах, а также разработке страницы выбора мест и оформления бронирования. Этот день был особенно насыщен работой с React-компонентами, взаимодействием с API, управлением состоянием и стилизацией через Ant Design.Утром я приступил к созданию страницы отдельного фильма. Используя useParams из react-router-dom, настроил динамическую маршрутизацию по ID фильма. Выполнил запрос на backend по маршруту /api/movies/{id} и получил все необходимые данные: название, описание, жанр, длительность, возрастной рейтинг, изображения, а также связанные сеансы. Для отображения использовал Card, Image, Descriptions, Tag и Button из Ant Design, благодаря чему интерфейс получился аккуратным и современным.Следующим шагом была реализация списка сеансов. У каждого фильма может быть несколько показов в разные даты и время. Я отобразил их в виде таблицы с кнопкой «Выбрать места» для каждого варианта. При нажатии на кнопку пользователь перенаправляется на страницу выбора мест. Здесь началась реализация самой важной части — интерфейса бронирования.На странице выбора мест я построил план зала кинотеатра в виде сетки с рядами и местами. Для этого я сгенерировал двумерный массив и визуализировал его с помощью сетки Row и Col от Ant Design. Используя стили, я визуально выделил свободные, занятые и выбранные пользователем места разными цветами. Данные о занятых местах приходят с backend и отображаются автоматически. При выборе пользовательских мест происходит обновление состояния, которое хранится в локальном состоянии компонента.Добавлена форма для подтверждения бронирования: поле для выбора количества билетов, автоматический расчёт суммы, поле для подтверждения и кнопка «Забронировать». После нажатия данные отправляются на backend через POST /api/bookings, включая ID сеанса и массив выбранных мест. При успешной отправке пользователь перенаправляется в личный кабинет, где он может просмотреть свои заказы.Также я реализовал простой способ авторизации для взаимодействия с API: в axios по умолчанию устанавливается Authorization заголовок с JWT-токеном, полученным при входе. Это позволило обезопасить маршруты и упростить последующую разработку защищённых разделов.  Было протестировано поведение приложения в случае отсутствия токена: запросы корректно блокируются. Под конец дня я начал верстку |  |
| *13.05.2025* | Седьмой день практики был посвящён расширению пользовательского функционала приложения и улучшению взаимодействия клиента с сервером. Основное внимание было уделено реализации личного кабинета пользователя, отображению истории заказов, возможности отмены бронирования, а также внедрению отзывов о фильмах. Кроме того, я начал оптимизацию структуры компонентов, вынос повторяющейся логики в отдельные модули и улучшение визуального оформления страниц с помощью Ant Design.Начал день с создания страницы «Личный кабинет». Эта страница доступна только авторизованным пользователям и позволяет им просматривать информацию о себе и своих бронированиях. Используя axios и useEffect, я сделал запрос на backend (GET /api/user/bookings) и отобразил полученные данные в таблице. Таблица построена с помощью компонента Table из Ant Design. В ней отображаются названия фильмов, дата и время сеанса, номера выбранных мест и статус брони (активна или отменена). Это позволило пользователю получить наглядную и структурированную информацию о своих посещениях.Добавлена функциональность отмены бронирования. Напротив каждого активного заказа размещена кнопка «Отменить», по нажатию на которую отправляется DELETE-запрос на сервер. Если операция успешна, происходит обновление данных на клиенте и выводится сообщение о том, что бронирование отменено. Для защиты от случайных действий реализовано подтверждение через модальное окно (Modal.confirm), что улучшает пользовательский опыт.В рамках личного кабинета я реализовал возможность оставлять отзыв на фильм после посещения сеанса. Пользователь видит список просмотренных фильмов, рядом — кнопка «Оставить отзыв». Открывается модальное окно с формой: поле для текста отзыва, компонент Rate для выставления оценки (1–5 звёзд) и кнопка отправки. Данные уходят на backend по маршруту POST /api/reviews, после чего отзыв сохраняется и отображается на странице фильма.Для повышения удобства работы я начал оптимизировать повторяющийся код. Создал сервис api.js, в котором описаны все запросы к серверу: getMovies, getUserBookings, createReview и т.д. Это позволяет сократить дублирование логики, централизовать обработку ошибок и легко обновлять базовый URL или заголовки. Также внедрил глобальный обработчик ошибок, чтобы при 401 (Unauthorized) пользователя автоматически направляло на страницу входа, а при 500 — выводилось сообщение о системной ошибке. Вечером я протестировал поведение личного кабинета на различных сценариях: вход без бронирований, пользователь с несколькими заказами. |  |
| *14.05.2025* | Восьмой день практики был посвящён реализации административной панели и функционала управления контентом внутри приложения «Кинотеатр». Админ-панель — ключевой элемент проекта, так как именно с её помощью администраторы смогут добавлять новые фильмы, редактировать расписание сеансов, просматривать заказы и отзывы пользователей. Моя задача состояла в создании удобного, функционального и безопасного интерфейса для управления данными с возможностью масштабирования в будущем.Утром я начал с проектирования архитектуры административной части. Решил реализовать её внутри того же frontend-приложения, но с доступом только для пользователей с ролью admin. В backend-приложении уже была настроена роль администратора и middleware для защиты маршрутов. На frontend я добавил защиту маршрутов: при попытке доступа к разделу /admin обычным пользователем происходит редирект на главную страницу и отображается уведомление о запрете доступа.Затем приступил к созданию интерфейса админ-панели. Используя Layout, Menu, Breadcrumb и PageHeader из библиотеки Ant Design, я создал структуру: слева — вертикальное меню навигации (Фильмы, Сеансы, Бронирования, Отзывы), справа — основная рабочая область. Интерфейс выполнен в минималистичном стиле, с упором на читаемость и удобство навигации.  Первым модулем стал раздел «Фильмы». Здесь отображается таблица всех фильмов, с возможностью редактирования и удаления. Реализована форма добавления нового фильма — с полями: название, жанр, длительность, возрастной рейтинг, описание, ссылка на постер и дата начала проката. Все поля валидируются на клиенте. При успешной отправке данные уходят на backend (POST /api/admin/movies), и таблица обновляется. Для редактирования используется модальное окно, где администратор может внести изменения и сохранить. Все действия сопровождаются всплывающими уведомлениями о результате. Далее реализовал модуль «Сеансы». В нём администратор может выбрать фильм и создать на него сеанс: указать дату, время начала, количество мест, стоимость билета. Все данные отправляются на сервер и записываются в таблицу sessions. Также добавлена возможность удаления и изменения сеансов, при этом проверяется наличие активных бронирований — если они есть, система не позволяет удалить сеанс и сообщает об этом. После завершения функциональной части я провёл обширное тестирование.  Проверил все маршруты на уровне прав доступа, протестировал формы на ошибки. |  |
| *15.05.2025* | Сегодняшний, девятый день практики, был посвящён улучшению пользовательского интерфейса, повышению доступности и внедрению адаптивной верстки в веб-приложении «Кинотеатр». Также была произведена работа по оптимизации клиентского кода, улучшению UX-компонентов, добавлению анимаций, редизайну некоторых элементов, а также обеспечению корректной работы приложения на мобильных устройствах. Это особенно важно, так как значительная часть пользователей заходит на сайт именно со смартфонов.  Первым делом я начал с анализа текущей структуры компонентов на наличие дублирующегося кода и устаревших решений. Некоторые элементы (например, кнопки, карточки, заголовки и формы) были реализованы вручную, без переиспользования. Я вынес такие элементы в отдельные универсальные компоненты (CustomButton, MovieCard, PageHeader) с параметрами и стилизацией, соответствующей общей теме сайта. Это дало возможность упростить работу с интерфейсом и обеспечить единообразный внешний вид на всех страницах.Затем приступил к адаптивной верстке. Используя систему сетки Ant Design (Row и Col) и useBreakpoint, я переработал ключевые страницы: главную, страницу фильма, бронирование, личный кабинет и админ-панель. Были реализованы разные схемы отображения в зависимости от ширины экрана: на десктопе карточки фильмов выводятся по 4 в ряд, на планшете — по 2, а на телефоне — по 1. Также адаптированы размеры шрифтов, отступы, кнопки и поля форм. Мобильное меню было заменено на Drawer (выпадающую панель сбоку), что значительно улучшило UX на маленьких экранах.Особое внимание уделил странице бронирования. План зала теперь корректно отображается в виде прокручиваемой сетки на телефонах. Реализована плавная анимация при выборе мест, а также подсказки при наведении и нажатии. Для визуализации состояния мест (свободно, занято, выбрано) использовал разные цвета и иконки. Это позволяет пользователю быстро ориентироваться даже при большом количестве мест.Я также улучшил работу с формами: добавил автозаполнение данных пользователя на страницах бронирования и отзыва, валидацию с подсказками и иконками. Использовал Ant Design Form.Item с кастомными сообщениями об ошибках. Также внедрил пользовательские уведомления: после успешного действия отображается notification.success, а при ошибке — notification.error. Это помогает пользователю всегда понимать, что происходит в системе. Таким образом, проект стал не только функциональным, но и удобным, визуально приятным |  |
| *16.05.2025* | Десятый день практики был посвящён комплексной SEO-оптимизации веб-приложения «Кинотеатр», а также первичной подготовке к размещению проекта в сети. Основной задачей было обеспечить соответствие сайта современным требованиям поисковых систем, повысить его видимость в результатах поиска и улучшить предварительный просмотр страниц при расшаривании в социальных сетях. Это важный этап, особенно если проект планируется для реального использования, так как SEO напрямую влияет на трафик и эффективность приложения.Утром я продолжил работу с библиотекой react-helmet, которую начал интегрировать накануне. Эта библиотека позволяет программно управлять содержимым тега <head> в React-приложениях. Для каждой страницы (главная, страница фильма, бронирование, личный кабинет и т.д.) я добавил динамически формируемые теги <title>, <meta name="description">, <meta name="keywords">, а также OpenGraph-теги: og:title, og:description, og:image, og:url. Теперь, например, при переходе на страницу фильма в <title> отображается «Кинотеатр – [Название фильма]», а описание включает жанр и краткий синопсис. Это улучшает кликабельность при отображении страницы в результатах поиска и социальных сетях.Затем я составил и добавил в проект файл robots.txt, в котором указал разрешение индексации всех страниц для поисковых ботов, кроме /admin и API-маршрутов. Также сгенерировал sitemap.xml, который включает ссылки на все основные разделы сайта и страницы фильмов. Файл был добавлен в корень проекта, и я протестировал его через инструменты Google Search Console и Yandex Webmaster Tools — они корректно считывают карту сайта и видят все страницы.Дополнительно я реализовал микроразметку Schema.org с помощью JSON-LD. В карточки фильмов и на страницу фильма добавлены блоки, описывающие структуру данных: @type: Movie, название, режиссёр, продолжительность, изображение и т.д. Эта микроразметка помогает поисковым системам лучше понимать контент страницы и отображать его в виде расширенных сниппетов.Для повышения производительности я провёл аудит через Google PageSpeed Insights. Первоначальные результаты показали проблемы: большие изображения, отсутствие кэширования и рендеринг блокирующих ресурсов. Я оптимизировал изображения, подключил шрифты через display=swap, добавил кэширование для статики и минифицировал CSS и JS. После этого баллы производительности выросли с 67 до 89 на мобильных устройствах и до 98 на десктопе. Это значительное улучшение, которое положительно скажется на скорости загрузки и ранжировании в поиске. |  |
| *17.05.2025* | Сегодняшний день практики был посвящён интеграции и тестированию ключевых компонентов фронтенд-приложения «Кинотеатр», разработанного с использованием React, TypeScript, Vite и Ant Design. Основное внимание было уделено обеспечению корректной работы пользовательского интерфейса, взаимодействию с backend-API и улучшению пользовательского опыта.Утро началось с настройки маршрутизации в приложении. С использованием react-router-dom были определены основные маршруты: главная страница, страница фильмов, сеансы, бронирование и личный кабинет пользователя. Это обеспечило плавную навигацию между различными разделами приложения без перезагрузки страницы.Далее была реализована аутентификация пользователей. С помощью контекста React (React.Context) и хука useContext была создана глобальная система управления состоянием аутентификации. При входе пользователя JWT-токен сохраняется в localStorage, и состояние пользователя обновляется во всём приложении. Это позволило ограничить доступ к определённым маршрутам и компонентам только для авторизованных пользователей.Особое внимание было уделено странице бронирования билетов. Здесь пользователь может выбрать фильм, сеанс и места в зале. Для отображения мест использовалась сетка (Grid) из Ant Design, где каждое место представлено интерактивной кнопкой. Занятые места отображаются в сером цвете и недоступны для выбора, свободные — в зелёном, а выбранные пользователем — в синем. Это обеспечивает наглядность и удобство при выборе мест.После выбора мест и подтверждения бронирования данные отправляются на сервер через axios. В случае успешного бронирования пользователь получает уведомление и может просмотреть свои заказы в личном кабинете. В случае ошибки отображается соответствующее сообщение, и пользователь может повторить попытку.Также была реализована страница личного кабинета пользователя. Здесь отображаются все бронирования пользователя с возможностью фильтрации по дате и фильму. Для каждого бронирования отображается информация о фильме, сеансе, выбранных местах и статусе бронирования. Это позволяет пользователю легко управлять своими заказами и при необходимости отменять бронирования.В завершение дня была проведена оптимизация производительности приложения. С использованием React.memo и useCallback были предотвращены ненужные перерисовки компонентов, что улучшило отзывчивость интерфейса. Также была проведена минификация и сборка проекта с помощью Vite, что снизило размер бандла и ускорило загрузку. |  |
| *19.05.2025* | Двенадцатый день практики был сосредоточен на завершении функционала пользовательской части, проведении итогового тестирования всего фронтенд-приложения «Кинотеатр» и исправлении багов, выявленных в процессе ручной проверки. Также я начал оформление пользовательского интерфейса с точки зрения логики навигации, интерфейсной целостности и базовой визуальной идентичности — чтобы приложение выглядело завершённым, логичным и удобным для восприятия.Первую половину дня я посвятил аудиту всех страниц, маршрутов и пользовательских сценариев. Проверил корректность переходов между страницами, отображение ошибок и уведомлений, а также доступность защищённых маршрутов только для авторизованных пользователей. Провёл ручное тестирование следующих сценариев: регистрация нового пользователя, вход в систему, переход к списку фильмов, выбор фильма, просмотр сеансов, выбор мест, бронирование, переход в личный кабинет, просмотр и отмена брони. На каждом этапе проверялись не только happy-path-сценарии, но и edge-case'ы: неправильный ввод данных, несуществующий ID фильма, попытка бронирования без выбора мест и так далее.По итогам тестирования были выявлены и устранены следующие баги: при повторном бронировании в одном сеансе система допускала выбор уже занятых мест — логика обновлена, теперь происходит дополнительная проверка статуса мест на момент отправки запроса;некорректная работа logout — при выходе не очищались некоторые поля состояния, из-за чего отображались старые данные;при изменении ширины экрана происходил сбой в отображении компонентов в личном кабинете — добавлены условия адаптивности;неинформативные ошибки при неудачном бронировании — обновлены обработчики ошибок и отображение пользовательских сообщений.Далее я занялся финальной стилизацией интерфейса. Был проанализирован весь визуальный стиль приложения: цветовая палитра, типографика, отступы и размеры компонентов. Все ключевые элементы (кнопки, карточки, формы, хедеры и футеры) были приведены к единому виду. Основным цветом выбрана глубокая тёмно-синяя палитра, акцентным — оранжевый для кнопок и действий. Typography был стандартизирован: заголовки — жирные, 20–24px, основной текст — 16px. Это улучшило восприятие и повысило визуальную иерархию элементов. К концу дня все баги были устранены, и интерфейс стал завершённым, эстетически выверенным и функционально полным. Сделал финальную проверку с точки зрения UX — на всех этапах |  |
| *20.05.2025* | На тринадцатый день практики я сосредоточился на тестировании, оптимизации и финальной доработке backend-части проекта «Кинотеатр» на Laravel, а также подготовке к полноценному развертыванию приложения на сервере. Этот этап был важен для обеспечения стабильной работы серверной логики, повышения безопасности и производительности API, а также для приведения всего backend-кода к завершённому, чистому состоянию, готовому к продакшен-среде.Утро началось с рефакторинга существующих контроллеров. Я проанализировал код на наличие дублирования, длинных методов и несоответствий принципам SOLID. Были выделены сервисные классы (BookingService, MovieService), в которые перенесена бизнес-логика: расчёт доступных мест, проверка статуса брони, создание сеанса и т.д. Это сделало контроллеры более читаемыми и облегчило их тестирование.Следующим этапом я провёл тестирование всех API-эндпоинтов с помощью Postman. Создал коллекцию всех маршрутов: авторизация, регистрация, получение фильмов, просмотр сеансов, создание брони, отмена брони, отправка отзывов и действия администратора. Протестировал корректность работы каждого маршрута на валидных и невалидных данных. Были проверены коды ответов, содержание тела ответа, валидация, обработка исключений и взаимодействие с базой данных.После этого я настроил централизованную систему логирования и обработки ошибок. В Laravel через Handler.php я добавил логирование всех необработанных исключений с указанием IP-адреса, ID пользователя (если есть), маршрута и тела запроса. Это позволит администратору в будущем отслеживать критические сбои и быстрее их устранять. Также все ошибки теперь возвращаются клиенту в едином формате с ключами status, message, errors, что удобно при обработке на фронтенде.Затем я приступил к аудиту безопасности. Проверил наличие CSRF-защиты, убедился, что все защищённые маршруты обернуты middleware auth:sanctum, и что доступ к административным маршрутам ограничен только пользователям с ролью admin. Для входа и регистрации добавлены rate-limit middleware, ограничивающие количество запросов с одного IP-адреса. Валидация данных усилена, все входящие данные фильтруются, а потенциально опасные символы — экранируются. Это минимизирует риск SQL-инъекций, XSS и других атак. К концу дня я проверил работу frontend-приложения в связке с обновлённым backend. Провёл тестирование авторизации, бронирования, отзывов, отмены и просмотра истории. Все данные передавались корректно, ответы от API были быстрыми. |  |
| *21.05.2025* | На четырнадцатый день практики я сосредоточился на тестировании готового веб-приложения «Кинотеатр» в связке frontend и backend, а также подготовке финального релиза проекта. Этот день стал завершающим в разработке и проверке основной функциональности перед развертыванием на сервере. Главной целью было провести ручную и автоматизированную проверку всех ключевых пользовательских сценариев, убедиться в стабильности работы приложения и подготовить итоговую демонстрацию. Первым делом я собрал фронтенд-приложение с помощью Vite: выполнил npm run build, чтобы получить оптимизированный и минифицированный код. Проверил структуру папки dist, удостоверился, что пути к статическим ресурсам настроены корректно. Параллельно на backend-части обновил зависимости, провёл миграции (php artisan migrate:fresh --seed), очистил кэш маршрутов и конфигурации, и включил режим production. Далее приступил к ручному тестированию. Было выполнено полное прохождение всех сценариев от имени обычного пользователя: Регистрация и вход в систему. Просмотр списка фильмов и переход на страницу фильма. Выбор сеанса и мест в зале. Оформление бронирования и отображение в личном кабинете. Отмена брони. Оставление отзыва и его отображение в публичной части. Выход из системы и попытка перехода на защищённые маршруты. аждый из этапов работал стабильно, API корректно обрабатывало все запросы, токены авторизации обновлялись, и ошибки не возникали. Интерфейс отображал все данные без задержек, а визуальные элементы (кнопки, формы, таблицы, списки) были адаптивными и отзывчивыми. Далее проверил административную часть: ход с правами администратора. Добавление, редактирование и удаление фильмов. Создание новых сеансов с указанием зала, времени и стоимости.  Просмотр заказов всех пользователей. Модерация отзывов.Завершил день коммитом git commit -m "Final QA, tests, production-ready release" и экспортом всей истории изменений. Приложение «Кинотеатр» прошло полный цикл проектирования, разработки и тестирования, и теперь готово к реальному использованию и демонстрации как полноценный учебный продукт. |  |
| *22.05.2025* | Пятнадцатый день практики был посвящён развертыванию проекта «Кинотеатр» на удалённый сервер и настройке всей инфраструктуры, необходимой для стабильной работы приложения в продакшене. Это включало деплой backend-приложения на Laravel, frontend-сборки на Vite, настройку веб-сервера Nginx, SSL-сертификатов, базы данных, а также обеспечение безопасности и производительности. Первым делом я выбрал хостинг для размещения приложения. Был использован сервер на базе Ubuntu 22.04, предоставляющий SSH-доступ, на котором предстояло вручную настроить окружение. Через терминал подключился к серверу, обновил пакеты командой sudo apt update && sudo apt upgrade, затем установил нужные компоненты: PHP 8.2, MySQL, Composer, Nginx, Node.js 18 и npm. Также были установлены пакеты php-mbstring, php-curl, php-mysql, php-xml, php-bcmath и другие, необходимые для корректной работы Laravel. После этого я склонировал репозиторий с backend-приложением на сервер и выполнил его настройку: Установил зависимости через composer install. Скопировал и отредактировал .env файл, указав реальные параметры базы данных, SMTP-почты и ключи. Сгенерировал ключ приложения через php artisan key:generate. Выполнил миграции и сиды: php artisan migrate --seed. Настроил очередь команд и планировщик задач Laravel через supervisord и cron. Далее я установил и настроил веб-сервер Nginx. Создал конфигурацию виртуального хоста, указывающую путь к публичной папке проекта /public, настроил маршрутизацию и перенаправления. Для защиты соединения я установил SSL-сертификат с помощью Certbot и Let's Encrypt, выполнив sudo certbot --nginx, благодаря чему сайт стал доступен по HTTPS. После успешной настройки backend, я приступил к деплою frontend-приложения. Склонировал frontend-репозиторий, выполнил npm install, затем npm run build. Сборка поместилась в папку dist, и я настроил Nginx так, чтобы на корневом маршруте отображалась именно эта сборка. Все маршруты React перенаправлялись на index.html — это обеспечило корректную работу SPA при прямом переходе по ссылкам. Проверил работу приложения: открыл сайт в браузере, прошёл весь сценарий от регистрации до бронирования и оставления отзыва. Всё работало стабильно, без ошибок. |  |
| *23.05.2025* | На шестнадцатый день практики я сосредоточился на тестировании производительности и стабильности работы приложения «Кинотеатр» после его размещения на сервере. Основной целью было убедиться, что система работает быстро, стабильно и предсказуемо в условиях приближённых к реальному использованию. Это включало как ручное тестирование, так и инструментальное измерение времени отклика, скорости загрузки страниц и отзывчивости интерфейса на различных устройствах.С утра я начал с измерения скорости загрузки сайта вручную с разных браузеров: Google Chrome, Mozilla Firefox и мобильной версии Safari. Проверил поведение сайта как на десктопных, так и на мобильных устройствах. Все основные страницы — главная, страница фильма, страница бронирования, личный кабинет и админ-панель — открывались за 1–2 секунды, без подёргиваний или задержек. Визуальные элементы загружались последовательно, и пользовательский интерфейс оставался отзывчивым.Для более точной оценки производительности я воспользовался инструментом Lighthouse, встроенным в Chrome DevTools. Он позволяет протестировать сайт по таким параметрам, как производительность, доступность, SEO и практика безопасности. По результатам тестирования, приложение получило следующие оценки: производительность — 92, доступность — 90, лучшие практики — 100, SEO — 95. Это свидетельствует о том, что сайт хорошо оптимизирован, работает стабильно и готов к приёму пользователей.Особое внимание я уделил времени первого взаимодействия (First Input Delay) и времени до полной загрузки контента (Time to Interactive). На всех страницах эти значения составляли менее 1.5 секунд, что соответствует хорошему пользовательскому опыту. Также я проверил, как работает интерфейс при слабом интернет-соединении (имитировал 3G через DevTools) — даже в таких условиях интерфейс оставался стабильным, и вся функциональность приложения была доступна.Параллельно я протестировал интерфейс на клик-нагрузку: быстрое многократное нажатие по кнопкам, попытки отправки повторных запросов и переходы между страницами. Приложение устойчиво реагировало на все действия, не возникало двойных отправок форм, багов в маршрутизации или визуальных сбоев.Проверил также кэширование: повторный заход на сайт с того же устройства происходил заметно быстрее — это говорит о корректной работе механизма кеша статики браузером и сервером. Статические ресурсы (JS, CSS, изображения) эффективно отдаются из кэша, что снижает нагрузку на сервер и улучшает скорость загрузки. |  |
| *24.05.2025* | На семнадцатый день практики я сосредоточился на проведении комплексного аудита безопасности веб-приложения «Кинотеатр», разработанного с использованием стека ReactJS (фронтенд) и Laravel + MySQL (бэкенд). Целью данного этапа стало выявление возможных уязвимостей и проверка соответствия системы базовым требованиям веб-безопасности. Это было особенно важно перед запуском в реальную эксплуатацию, чтобы минимизировать риски атак и утечек данных пользователей. Первым делом я начал с анализа безопасности на клиентской стороне. Проверил, не содержатся ли в React-компонентах уязвимости, связанные с внедрением вредоносного кода (XSS). Убедился, что нигде не используется dangerouslySetInnerHTML без предварительной валидации или очистки данных. Все пользовательские данные, отображаемые в интерфейсе, проходят экранирование и не приводят к выполнению скриптов. Также была проверена работа с куками — они не используются для хранения чувствительной информации, а вся авторизация работает через токены (JWT), которые хранятся в HttpOnly куках или безопасном хранилище. Проверил конфигурации .env и убедился, что параметры APP\_DEBUG и APP\_ENV отключены в боевой версии, что предотвращает утечку стеков ошибок. Приложение не отдаёт чувствительные данные при сбоях, а исключения обрабатываются корректно через глобальный обработчик. Провёл ручной анализ маршрутов — все защищённые разделы (например, админ-панель, управление бронированиями) доступны только при наличии корректного токена авторизации и соответствующих прав доступа. Также реализована защита от CSRF на формах и маршрутах, использующих POST, PUT и DELETE. Особое внимание я уделил валидации входных данных на сервере. Проверил, что все пользовательские данные проходят через кастомные Laravel Request-классы с чётко заданными правилами валидации. Это исключает возможность SQL-инъекций и предотвращает некорректную запись в базу данных. Провёл ручное тестирование API с использованием Таким образом, проведённый аудит показал, что веб-приложение «Кинотеатр» соответствует базовым требованиям по безопасности и может быть развёрнуто в реальных условиях без риска критических уязвимостей. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Руководитель практики от колледжа:

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ | Пашкина Л.В., преподаватель |
| *подпись* | *Ф.И.О., должность* |