**Министерство образования Иркутской области**

Государственное бюджетное профессиональное образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| **ДП.09.02.07-1.24.202.15 ПЗ** | УТВЕРЖДАЮ  Зам. директора по УР, к.т.н.  \_\_\_\_\_\_\_\_\_\_Е.А. Коробкова |

**ВЕБ-ПРИЛОЖЕНИЕ «ПЛАНЕТАРИЙ»**

|  |  |  |
| --- | --- | --- |
| Нормоконтролер: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Е.С. Кудрявцева) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.И. Тирский) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Д.А. Отчесов) |

Иркутск 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc169867064)

[1 Описание предметной области 5](#_Toc169867065)

[2 Анализ инструментальных средств разработки, используемых при реализации программного продукта 8](#_Toc169867066)

[3 Проектирование программного продукта 12](#_Toc169867067)

[3.1 Структурная схема программного продукта 12](#_Toc169867068)

[3.2 Функциональная схема программного продукта 14](#_Toc169867069)

[3.3 Проектирование базы данных 16](#_Toc169867070)

[3.4 Проектирование интерфейса 20](#_Toc169867071)

[4 Разработка программного продукта 24](#_Toc169867072)

[4.1 Разработка базы данных программного продукта 24](#_Toc169867073)

[4.2 Разработка кода клиентской части программного продукта 25](#_Toc169867074)

[4.3 Разработка кода серверной части программного продукта 27](#_Toc169867075)

[4.4 Тестирование программного продукта 31](#_Toc169867076)

[5 Документирование программного продукта 37](#_Toc169867077)

[5.1 Руководство пользователя 37](#_Toc169867078)

[ЗАКЛЮЧЕНИЕ 40](#_Toc169867079)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42](#_Toc169867080)

[Приложение Б – Листинг главной страницы 45](#_Toc169867081)

ВВЕДЕНИЕ

Планетарий — научно-просветительное учреждение, в котором демонстрируется небесная сфера со звёздами, планетами и спутниками, кометами и метеорами; также солнечные и лунные затмения, панорамы Луны, Марса, Венеры и климатических поясов земного шара.

Планетарии могут быть как открытыми для публики, так и закрытыми, предназначенными для проведения научных исследований и образования.

В настоящее время планетарии являются важным инструментом для популяризации науки и повышения интереса к астрономии среди молодежи. Они помогают людям лучше понимать нашу вселенную и ее законы, а также помогают развивать научное мышление и воображение.

Актуальность данной темы заключается в том, что в наше время все больше и больше людей начинают увлекаться тематикой космоса. Данное приложение может помочь в освоении и получении необходимой информации, касающейся космоса.

Веб-приложение «планетарий» является удобным инструментом для мониторинга записей на сеансы для дальнейшего посещения планетария, освоении новой информации. Оно позволяет упростить процесс поиска сеансов, а также добавлять новые сеансы, с названием и временем.

Основным преимуществом веб-приложения планетарий является его удобство и автоматизация процесса учёта. Администраторы могут быстро и легко вносить данные о новых сеансах в систему, а посетители своевременно узнавать о новых сеансах и времени их проведения.

Цель дипломного проекта — разработать веб-приложение, в котором посетители и сотрудники смогут взаимодействовать с данными планетария.

Основными задачами данной работы являются:

* изучить деятельность веб-приложения «планетарий»;
* спроектировать базу данных;
* спроектировать пользовательский интерфейс;
* написать серверную часть для связки веб-приложения и базы данных;
* написать код веб-приложения;
* провести тестирование программного продукта;
* написать руководство пользователя.

1. Описание предметной области

В качестве предметной области выбрано научно-просветительное учреждение «Планетарий», который занимается просвещающей деятельностью в сфере астрономии и космонавтики.

Планетарий – научно-просветительное учреждение, в котором демонстрируется небесная сфера со звездами, планетами и спутниками, кометами и метеорами; также солнечные и лунные затмения, панорамы Луны, Марса, Венеры и климатических поясов земного шара. Демонстрация проводится при помощи специального прибора «Планетарий», сопровождается лекциями по астрономии, космонавтики и науке о Земле. Как правило, в планетариях изображения небесные объектов проецируются на большой куполообразный экран.

Классический планетарий представляет собой сложный оптико-механический прибор, состоящий из оптических линз, источников света и пластин с крошечными, размером в доли миллиметра, отверстиями. Перемещая линзы и пластины определённым образом, можно варьировать местоположение разных звёзд, создавая картины неба, соответствующие разным временам года и разным географическим координатам.

Цифровые проекторы существенно расширили возможности планетариев. Они могут использоваться как в комплексе с оптико-механическими системами, так и самостоятельно. Сегодня такие установки имеются в большинстве планетариев России.

Планетарии могут быть как открытыми для публики, так и закрытыми, предназначенными для проведения научных исследований и образования.

В настоящее время планетарии являются важным инструментом для популяризации науки и повышения интереса к астрономии среди молодежи. Они помогают людям лучше понимать нашу вселенную и ее законы, а также помогают развивать научное мышление и воображение.

Важнейшими составляющими для любого планетария являются:

* полукруглый купол, имитирующий небесную сферу;
* проекционная установка, посылающая световые лучи на купол для получения изображений звёзд, планет и других небесных тел.

Обслуживание посетителей планетария осуществляется путем проведения экскурсий гидом. Они проводят виртуальные путешествия по космическим объектам, рассказывают о созвездиях, планетах и других интересных астрономических явлениях.

В данной предметной области документооборот включает в себя информацию о следующих компонентах:

* Персонал.
* Сеанс.
* Посетители.

Гиды отвечают за контроль и обслуживание билетов у посетителей. Каждый билет, выданный посетителю, имеет следующие параметры:

* Название сеанса.
* Дата сеанса.
* Номер билета.
* Имя посетителя.
* Название планетария.
* Адрес планетария.

При работе с системой экскурсовод должен иметь возможность решать следующие задачи:

* Регистрировать каждый новый сеанс, указав его название и время проведения;
* Проводить экскурсии для посетителей;
* Следить за порядком и дисциплиной.

Дипломный проект начинается с определение предметной области и выбора области исследования. Затем выполняется сбор материала. После чего осуществляется публикация и защита работы.

Функциональные возможности:

* Ведение базы данных (запись, чтение, модификация и удаление);
* Обеспечение логической непротиворечивости базы данных;
* Реализация наиболее часто встречающихся запросов в готовом виде.
* Готовые отчеты.

Готовые запросы: Получение информации о билете.

1. Анализ инструментальных средств разработки, используемых при реализации программного продукта

Для решения поставленной задачи необходимо провести анализ инструментов для проектирования и разработки программного продукта с базой данных.

Инструменты для проектирования приложения представлены в таблице 1.

Таблица 1 – Инструменты для проектирования

|  |  |  |  |
| --- | --- | --- | --- |
| Инструмент | Интерфейс | Платформа | Добавление своих элементов |
| Draw.io | Простой и интуитивно понятный интерфейс | Онлайн | + |
| Figma | Простой и интуитивно понятный интерфейс | Онлайн | + |
| Visio | Простой и интуитивно понятный интерфейс | Десктоп | - |
| Dia | Простой в использовании интерфейс | Десктоп | - |
| LibreOffice Draw | Простой в использовании интерфейс | Десктоп | - |

Проектирование является одним из важнейших этапов в разработке веб-приложения, поэтому нужно выбрать удобный набор инструментов, для этих целей была создана таблица и выбраны три критерия оценки, а именно «Интерфейс», «Платформа», возможность «Добавление своих элементов». На основе этих параметров был выбран Figma, так как имеется возможность добавления своих фигур, элементов и иконок, и он не нуждается в установке.

Для разработки серверной части необходимо рассмотреть несколько языков программирования. В таблице 2 представлен сравнительный анализ актуальных и удобных языков программирования, подходящих для реализации поставленных задач.

Таблица 2 – Инструменты для разработки

|  |  |  |  |
| --- | --- | --- | --- |
| Название языка программирования | PHP | Ruby | Python |
| Простота обучаемости | + | – | + |
| Читабельность | + | – | + |
| Динамическая типизация | + | + | + |
| Интеграция баз данных | + | + | – |
| Объектно–ориентированные возможности | + | + | + |

По результатам анализа был выбран PHP, потому что он прост для обучения, читабелен, поддерживает интеграцию с базой данных (далее БД) без сторонних библиотек.

Необходимо изучить систему управления базами данных в которой будет реализовано хранение данных приложения. Для этого была составлена Таблица 3.

Таблица 3 – Сравнение СУБД для разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| Название СУБД | Oracle | MySQL | PostgreSQL |
| Язык программирования | С, С++, Java, Ruby, Objective | Delphi, C, Java, Лисп, Perl, PHP, Python, Ruby, SQL | pgSQL, PHP, Python, Ruby и др. |
| Исходный код | закрытый | открытый | открытый |
| Лицензия | коммерческая | свободная | свободная |

В результате анализа была выбрана MySQL, потому что она отлично подходит для веб-разработки, имеет достаточную безопасность и функционал, легко масштабируется и поддерживает большой объем данных. Имеет хорошо сконструированную систему управления пользователями для предоставления различных разрешений, а также имеет высокую скорость работы, поддерживает большую часть функционала SQL.

Для управления сервера MySQL, был выбран PHPMyAdmin. В PHPMyAdmin предоставляется возможность создавать базы данных, выполнять различные SQL запросы, создавать пользователей, которые будут в дальнейшем управлять вашими базами данных, экспортировать и импортировать базы данных. Но основная задача будет вставлять записи, удалять их, изменять, создавать базы данных. Интерфейс лаконичен и понятен без дополнительного инструктажа.

Для работы с базами данных было проведено сравнение соответствующих средств реализации баз данных, представленное в таблице 4.

Таблица 4 – Сравнение инструментов для работы с базами данных

|  |  |  |
| --- | --- | --- |
| Название инструмента | MySql  Workbench | phpMyAdmin |
| Бесплатность | + | + |
| Графическое представление | + | - |
| Управление без кода sql | + | + |
| Удобство | + | - |
| Механизм по созданию связей между полями таблиц | + | - |

Проведя анализ инструментов для работы с базами данных были выбраны оба варианта – Workbench за его удобство и функциональные возможности, а именно графическое представление. PhpMyAdmin для быстрого управления базой непосредственно из браузера.

В качестве IDE средства разработки, было принято решение использовать VS Code, потому что он прост и удобен, а также обладает необходимым набором функций и поддерживает необходимые для разработки библиотеки, а также веб разработку.

Для разработки веб-приложения был выбран язык Java, потому что он прост в обучении, имеет огромное количество готовых библиотек, Приложения, созданные с помощью Java, легче и компактнее по сравнению с их аналогами на основе Kotlin. В качестве среды разработки будет использоваться VS Code за счёт своей мультифункциональности, расширяемой библиотеки дополнений и готовых решений, а также его обширного количества настроек.

# Проектирование программного продукта

# Структурная схема программного продукта

Одним из важнейших этапов разработки является проектирование диаграмм, которые помогут лучше понять структуру приложения и его работоспособность в целом.

Диаграмма вариантов использования, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы, благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат.

На рисунке 1 изображена Use Case View, которая показывает структурную схему приложения «Планетарий» для пользователей.

Схема отображает действия, выполняемые пользователем и приложением.

Пользователь является актером. После запуска приложения пользователю предоставляется страница регистрации/авторизации, на которой пользователь имеет возможность авторизоваться в приложении, если у него уже имеется аккаунт, либо зарегистрировать аккаунт, если пользователь не имеет такового. После прохождения аутентификации, пользователю представляется список доступных сеансов, которые будут проходить в ближайшее время в планетарии. Пользователь может оформить билет на заинтересовавший его сеанс, благодаря которому он сможет попасть на сам сеанс в планетарии. Сам билет будет иметь информацию по предстоящему сеансу, а именно: название сеанса, цена за билет, номер билета, дата начала и дата окончания сеанса. Билет необходим для того, чтобы попасть на сеанс. Будет отдельный раздел, в котором будут находиться вкладки с информацией, касающейся астрологии, с которой пользователь может ознакомиться.

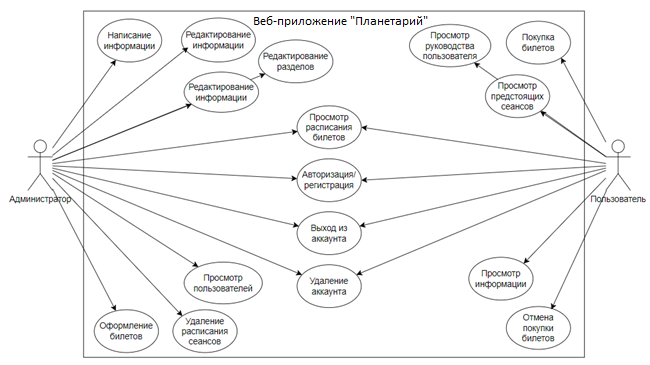


Рисунок 1 – Диаграмма прецедентов

Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

В итоге проектирования диаграммы деятельности были выделены основные возможные действия пользователя с программным продуктом.

Веб-приложение «Планетарий» имеет сложную структуру, состоящую из множества различных компонентов и сервисов. Для того чтобы отобразить диаграмму деятельности такого приложения, необходимо сначала определить основные компоненты и их взаимосвязи, а затем представить их в виде диаграммы.

Основные компоненты веб-приложения «Планетарий»:

– Серверная часть: включает в себя серверы, на которых работает приложение, базы данных, в которых хранятся данные пользователей и другая информация, необходимая для работы приложения.  
– Клиентская часть: это веб-страницы, которые пользователи видят в своих браузерах, а также мобильные приложения, если они есть.  
– API: это интерфейс, через который клиентская и серверная части взаимодействуют друг с другом.  
– Сервисы: различные сервисы, которые могут использоваться приложением, например, сервисы геолокации, определения местоположения пользователя, сервисы для работы с картами и так далее.

# Функциональная схема программного продукта

После определения основных компонентов можно построить диаграмму деятельности, которая покажет взаимосвязи между этими компонентами и как они взаимодействуют друг с другом. Диаграмма деятельности — это UML-диаграмма, которая показывает действия и их состояния. Она состоит из узлов (прямоугольники, ромбы, широкие полосы) и стрелок, которые показывают потоки управления и объекты данных. Используется для моделирования бизнес-процессов, технологических процессов и последовательных и параллельных вычислений. Диаграмма деятельности используется в проектировании веб-приложений для отображения пошаговых действий, которые должна выполнить система после инициирования прецедента, позволяет детально описать алгоритмическую и логическую реализацию прецедентов. Такая диаграмма поможет понять, какие части приложения наиболее критичны для его работы и какие проблемы могут возникнуть при изменении или обновлении одной из частей.

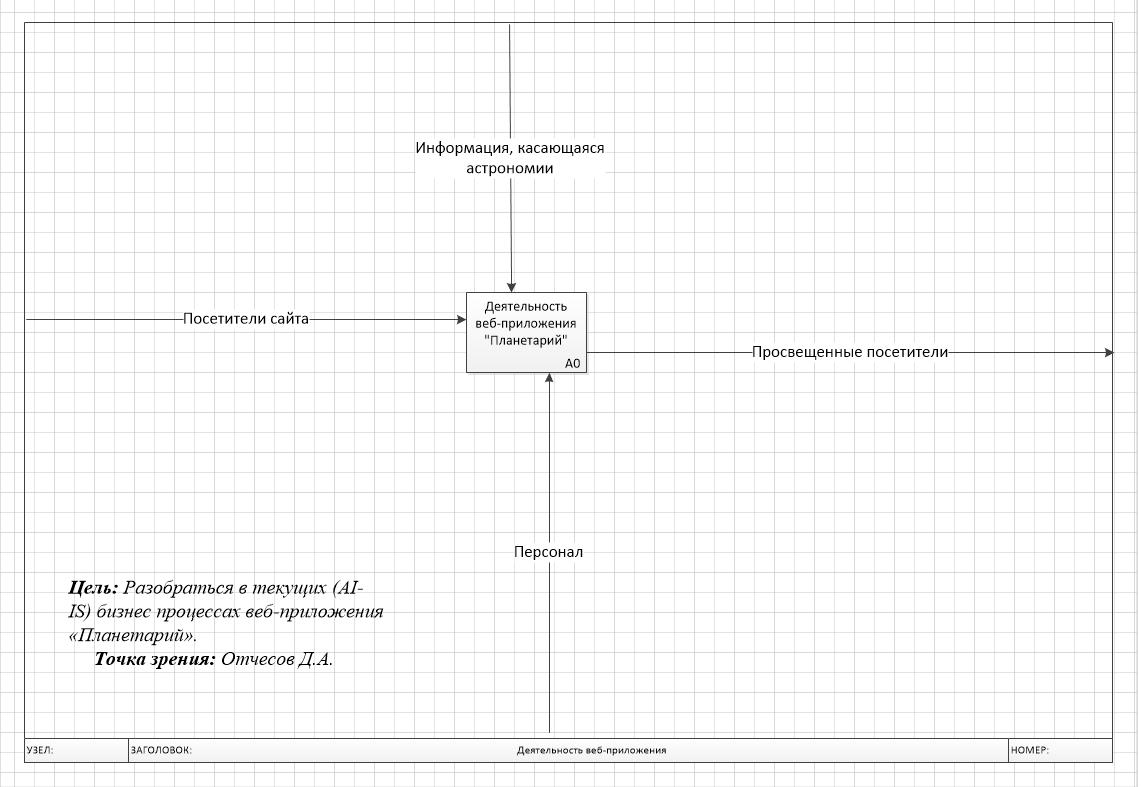


Рисунок 2 – Контекстная диаграмма

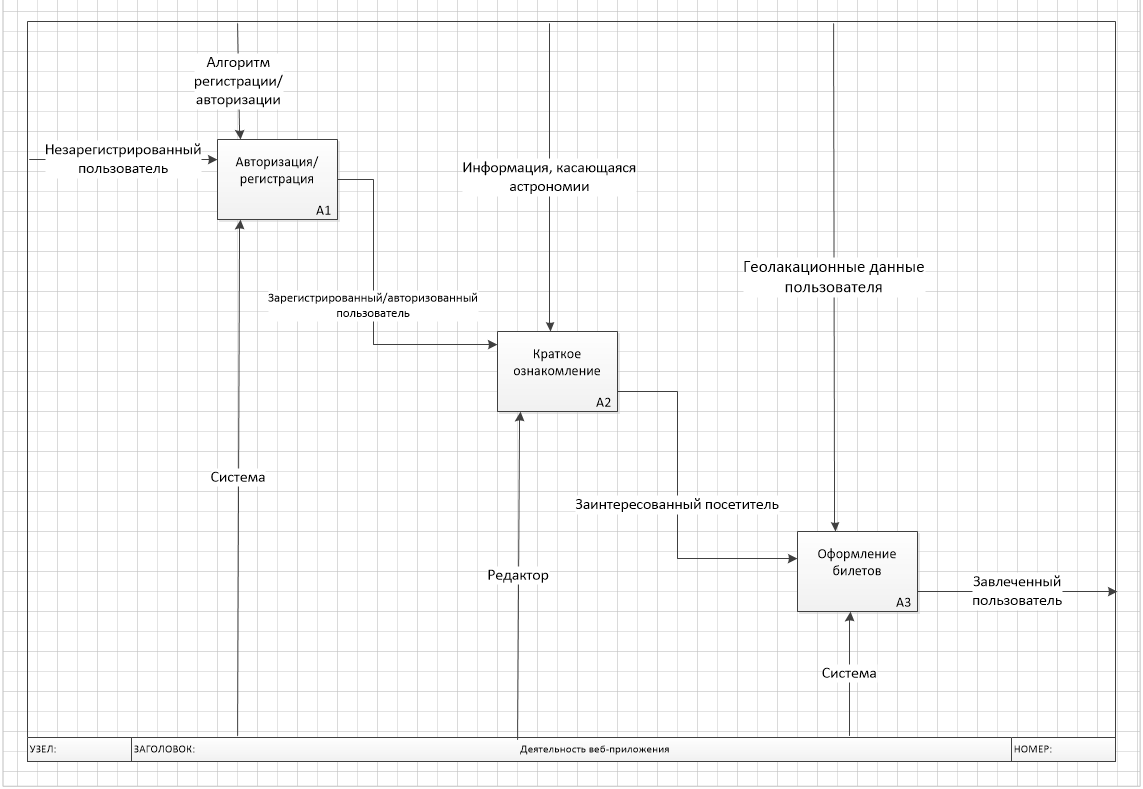


Рисунок 3 – Диаграмма деятельности

# Проектирование базы данных

Инфологическая модель базы данных представляет собой описание структуры и содержания данных, а также связей между ними, на уровне абстракции, без учета конкретных особенностей реализации. Она описывает данные в терминах объектов и связей между ними, без привязки к конкретной модели данных или технологии. Инфологическая модель используется для анализа и проектирования баз данных, а также для разработки концептуального уровня архитектуры информационных систем.

На рисунке 4 представлена инфологическая модель базы данных для веб-приложения «Планетарий».

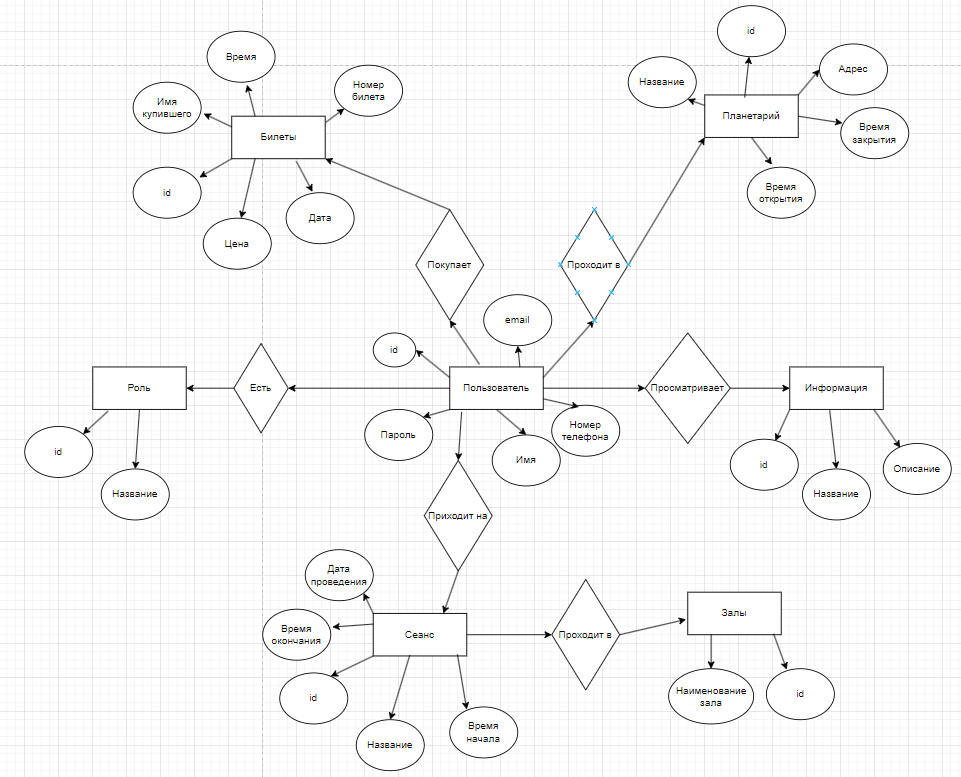


Рисунок 4 – Инфологическая модель базы данных

ER (Entity-Relationship) модель базы данных – это графический способ описания структуры данных, который представляет собой набор сущностей (entities) и связей (relationships) между ними. ER модели используются для анализа и проектирования баз данных и помогают определить необходимые сущности, атрибуты и связи для создания эффективной и корректной базы данных.

На рисунке 5 представлена ER модель базы данных для веб-приложения «Планетарий». В атрибутах таблицы «user» указаны персональные данные для регистрации и последующей авторизации в веб-приложении. Атрибуты таблицы «user» включают персональные данные, такие как имя пользователя, логин и пароль для авторизации, а также номер телефона, чтобы иметь возможность быстро с связаться с администратором. Сеансы будут отображаться на главной странице.

На модели представлено шесть сущностей, которые имеют свои уникальные атрибуты. Также сущности связаны между собой и в сумме получается структура под названием – инфологическая модель. Инфологическая модель – это модель, которая отражает семантику предметной области и структуру базы данных. Она помогает определить, как взаимодействуют сущности, то есть объекты реального мира в системе.

Проектирование базы данных не ограничивается только этим этапом разработки, а проходит на протяжении всей разработки, до того момента пока в системе не появятся данные, которые нельзя потерять. Результатом проектирования базы данных является ER-модель. ER-модель – это модель базы данных, используемая при высокоуровневом, или же концептуальном, проектировании баз данных. Она позволяет выделять ключевые сущности и обозначить связи между ними. Для визуализации ER-моделей используют диаграмму «сущность-связь». Она используется в разработке программного обеспечения для визуализации и анализа взаимосвязей между сущностями и атрибутами системы.

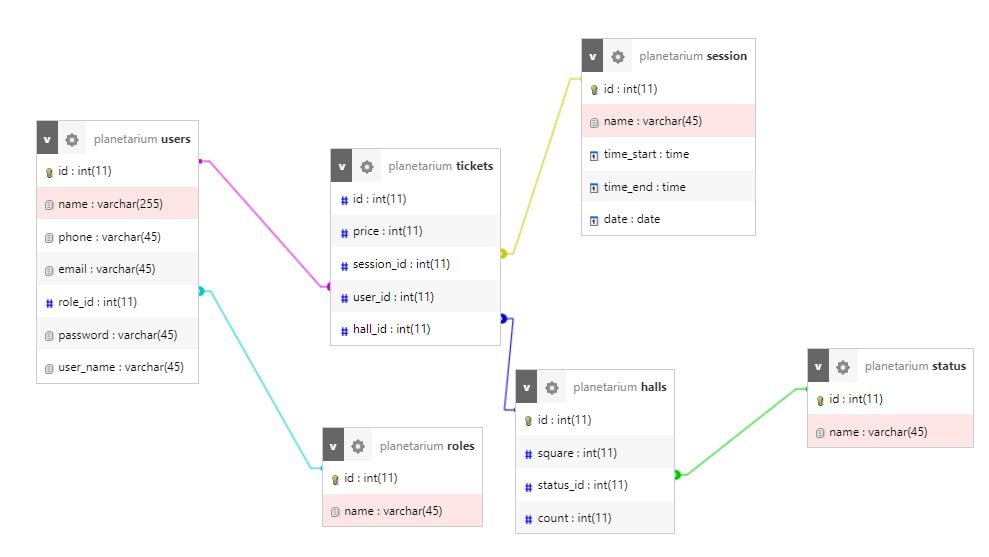


Рисунок 5 – ER-модель базы данных

Всего она содержит 6 таблиц:

Таблица 5 – Таблица «Билеты»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | Int(11) | Идентификатор |
| Цена | int(11) | Цена билета |
| session\_id | int(11) | Идентификатор сессии |
| user\_id | int(11) | Идентификатор пользователя |
| hall\_id | int(11) | Идентификатор зала |

Таблица 6 – Таблица «роли»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | int(11) | Идентификатор |
| Название | varchar(45) | Название роли |

Таблица 7 – Таблица «залы»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | int(11) | Идентификатор |
| Status\_id | int(11) | Идентификатор статуса |
| Square | int(11) | Площадь |

Таблица 8 – Таблица «статус»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | int(11) | Идентификатор |
| Название | varchar(45) | Фамилия администратора |

Таблица 9 – Таблица «сессии»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | Int(11) | Идентификатор |
| Название | varchar(45) | Название сеанса |
| Время начала | time | Время начала сеанса |
| Время конца | time | Время конца сеанса |
| Дата | date | Дата проведения сеанса |

Таблица 10 – Таблица «пользователи»

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ID | Int(11) | Идентификатор |
| Имя | varchar(255) | Имя пользователя |
| Номер телефона | varchar(45) | Номер телефона пользователя |
| Почта | varchar(45) | Почтовый адрес пользователя |
| Пароль | varchar(45) | Пароль пользователя от аккаунта |
| role\_id | int(11) | Идентификатор роли |

Сама база данных была доведена до второй нормальной формы. Но прежде, чем она стала таковой пришлось пройти два этапа, это:

1. Нормализация до первой нормальной формы.
2. Нормализация до второй нормальной формы.

Далее рассмотрим, что такое нормальная форма. Нормальная форма– свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, которая потенциально может привести к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение. Существует шесть нормальных форм, и имеющаяся база данных была доведена до второй.

При нормализации в первую нормальную форму были соблюдены условия, которые делают ее таковой. А именно все данные должны быть атомарными. То есть одно поле – одно значение. Это условие было соблюдено и таблица в первой нормальной форме, идем дальше.

Для нормализации во вторую нормальную форму необходимо: во-первых, она уже должна быть в первой нормальной форме, а во-вторых, таблицы должны быть связаны между собой, для того чтобы данные в таблицах не дублировались. Ведь если иметь одинаковые записи в разных таблицах, то придется изменять данные и в одной, и в другой таблице.

Таким образом, можно увидеть всю необходимую информацию для понимания системы хранения данных.

# Проектирование интерфейса

Пользовательский интерфейс – это средства взаимодействия между человеком и компьютером. Говоря простыми словами, интерфейс – внешняя часть программы или устройства, с которыми работает пользователь.

Интерфейсы являются основой взаимодействия всех современных приложений. Если интерфейс какого-либо объекта не изменяется (стабилен, стандартизирован), это даёт возможность модифицировать сам объект, не перестраивая принципы его взаимодействия с другими объектами. Интерфейсы играют ключевую роль в создании веб-приложений, обеспечивая согласованность кода, упрощая тестирование и повышая гибкость и масштабируемость программного обеспечения.

Данный прототип интерфейса был построен в онлайн конструкторе Figma.

Для дальнейшей работы с сайтом, пользователю необходимо будет зарегистрироваться или же авторизоваться на сайте. Если же у пользователя все же есть аккаунт, пользователь должен ввести данные, которые должны быть известны только ему. На рисунке 6 представлен прототип страницы авторизации, где будут вводиться данные от аккаунта.

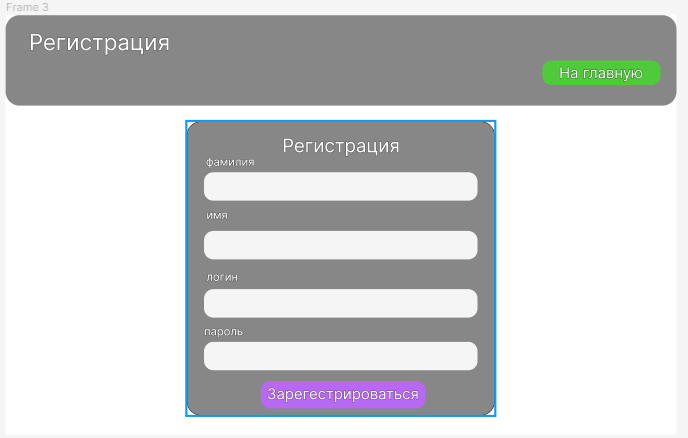


Рисунок 6 – Прототип страницы «Регистрации»

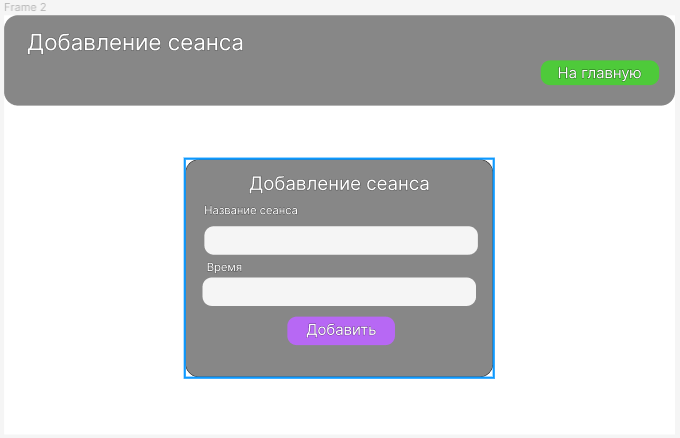


Рисунок 7 – Прототип страницы «Добавления сеанса»

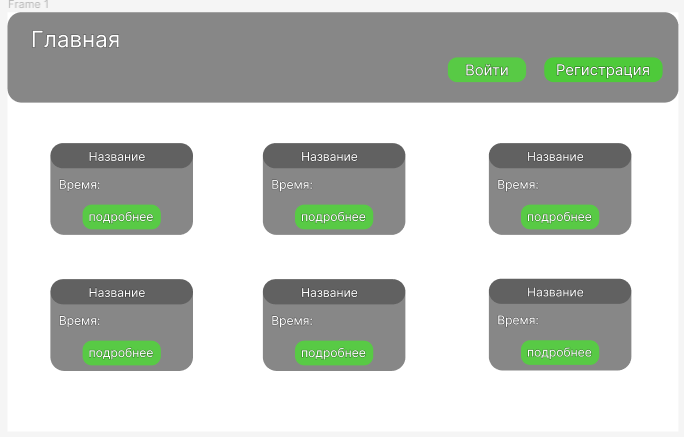


Рисунок 8 – Прототип страницы «Сеансы»

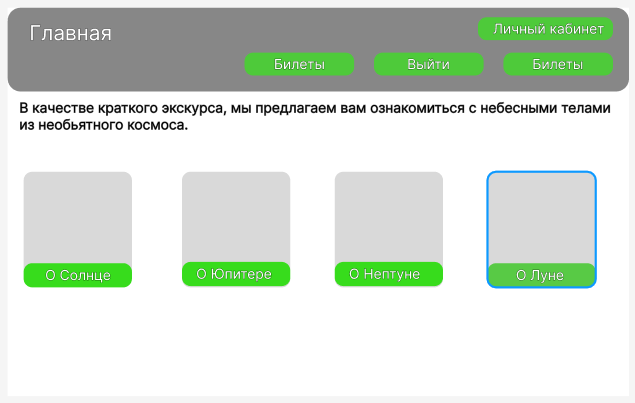


Рисунок 9 – Прототип страницы «Главная»



Рисунок 10 – Прототип страницы «Личный кабинет»

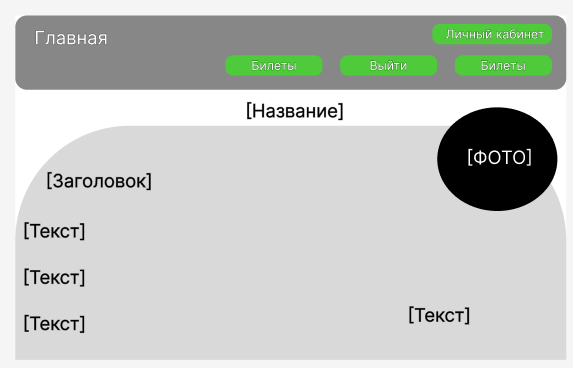


Рисунок 11 – Прототип страницы «Информация по планетам»

1. Разработка программного продукта
   1. Разработка базы данных программного продукта

Этап разработки базы данных программного продукта включает определение требований, проектирование, разработку и тестирование, интеграцию с продуктом, тестирование и отладку, а также развертывание и сопровождение.

Разработка базы данных веб-приложения реализовывалась в веб-приложении PhpMyAdmin. База данных состоит из шести таблиц.

Структуры таблиц соответствуют схеме проектирования базы данных из пункта 3.3.

Таблица «Залы» представлена на рисунке 12.



Рисунок 12 – Таблица «Залы»

Таблица «Роли» представлена на рисунке 13.

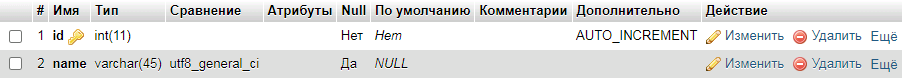


Рисунок 13 – Таблица «Роли»

Таблица «Статус» представлена на рисунке 14.

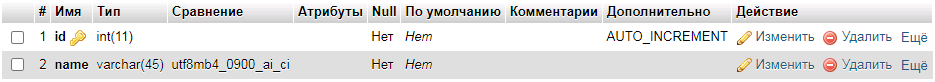


Рисунок 14 – Таблица «Статус»

Таблица «Сеансы» представлена на рисунке 15.



Рисунок 15 – Таблица «Сеансы»

Таблица «Билеты» представлена на рисунке 16.

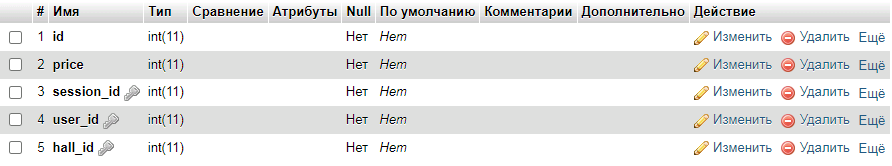


Рисунок 16 – Таблица «Билеты»

Таблица «Пользователи» представлена на рисунке 17.



Рисунок 17 – Таблица «Пользователи»

# Разработка кода клиентской части программного продукта

Разработка клиентской части программного продукта включает создание пользовательского интерфейса, который предоставляет возможность взаимодействия пользователя с приложением. Это включает в себя разработку дизайна интерфейса, создание элементов управления, таких как кнопки, поля ввода, списки и т.д. Также разработка клиентской части включает написание кода на языке программирования, который обрабатывает действия пользователя и отправляет запросы на сервер. Кроме того, клиентская часть должна быть интегрирована с серверной частью приложения, чтобы обеспечить взаимодействие между пользователем и сервером.

Клиентская часть веб-приложения «Планетарий» была написана на языке разметки гипер-текста HTML, а так же некоторый функционал был реализован на языке программирования JS.

Листинг главной страницы представлен в приложении Б. В листинге изображён полный код главной страницы, включая прописанные стили, подключение навигационной панели, в зависимости от роли авторизованного пользователя, а так же сама разметка. Навигационная панель на главной странице формируется в зависимости от авторизации пользователя, что реализовано через переменную сессии, а так же от роли пользователя – администратор, либо обычный пользователь.

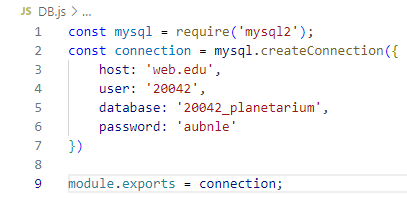


Рисунок 18 – Код подключения к БД

# Разработка кода серверной части программного продукта

Разработка серверной части – это создание и оптимизация компонентов, которые обрабатывают и хранят данные, обеспечивают безопасность, взаимодействуют с клиентом. Она включает выбор технологии и языка, разработку архитектуры, создание базы данных и реализацию логики приложения.

Серверная часть веб-приложения «Планетарий» была написана на языке программирования JS.

Подключение к базе данных осуществляется в отдельном файле, после чего подключается на страницах при помощи функции подключения. Код подключения к базе данных показан на рисунке 19.

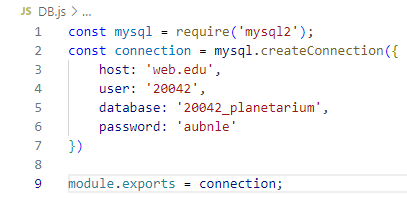


Рисунок 19 – Код подключения к базе данных

Для того, чтобы получить доступ к определённым функциям веб-приложения, таким как добавление сеансов и вход в личный кабинет, пользователю необходимо зарегестрироваться, либо авторизоваться.

При регистрации, введённые на странице регистрации данные, отправляются в обработчик скрипта, а затем в базу данных «users», так как преподавателей могут добавлять только администраторы. В самом файле обработки пароль хэшируется в кодировке sha256 для повышения безопасности, а так же выполянется проверка на правильность формата введённой электронной почты. Код регистрации пользователя представлен на рисунке 20.

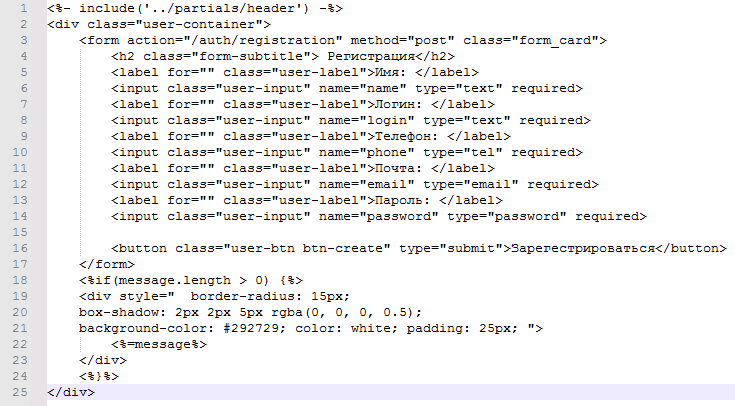


Рисунок 20 – Код регстрации пользователя

В случае, если у пользователя имеется учётная запись, необходимо пройти авторизацию. После того, как данные введены и роль выбрана, данные отпарвляются в скрипт для обработки, где происходит сверение данных в соответствующей выбранной роли таблице. Если совпадение найдено, все данные пользователя заносятся в переменную сессии и пользователь получает доступ к соответствующим роли функциям. Код авторизации пользователя представлен на рисунке 21.

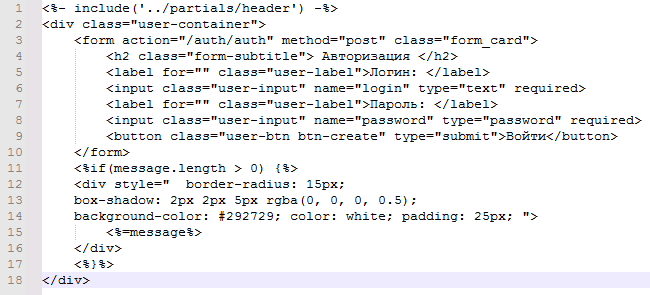


Рисунок 21 – Код авторизации пользователя

Если условия регистрации не будут соблюдены, а именно, длина пароля будет менее 6 символов, или такой пользователь уже существует, приложением будет выдана ошибка. Код данной ошибки представлен на рисунке 22.

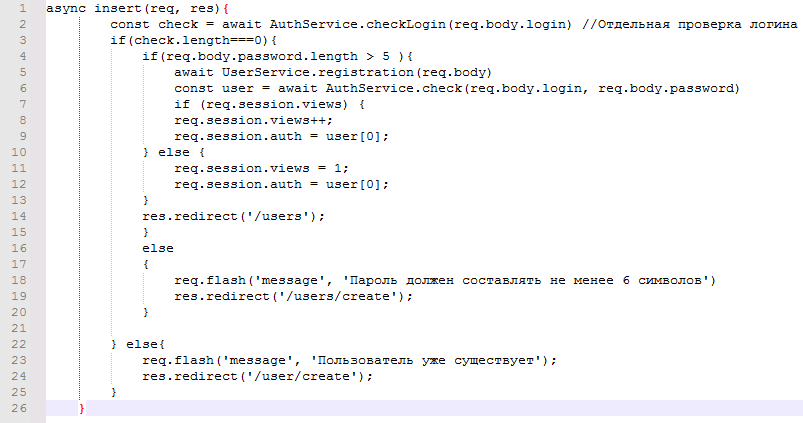


Рисунок 22 – Код выдачи ошибки при регистрации

Обладающий определенными правами пользователь имеет возможность добавлять сеансы для посещения обычным пользователям. Код создания сеанса представлен на рисунке 23.



Рисунок 23 – Код добавления сеанса

Если же информация по сеансу была занесена с ошибками, администратор всегда может изменить её в приложении. В базу данных заносятся название, идентификатор сеанса, время начала и время окончания, а также дата. Код добавления редактирования сеанса представлен на рисунке 24.

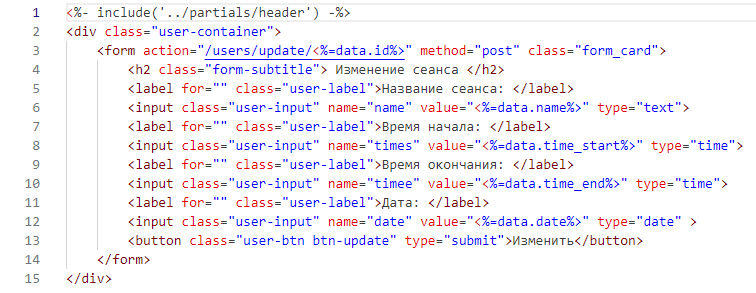


Рисунок 24 – Код редактирования сеанса

Обладая нужными правами администатора, пользователь может добавлять пользователей, купивших билет. Для этого необходимо ввести имя, логин и телефон пользователя. Код добавления пользователя представлен на рисунке 25.

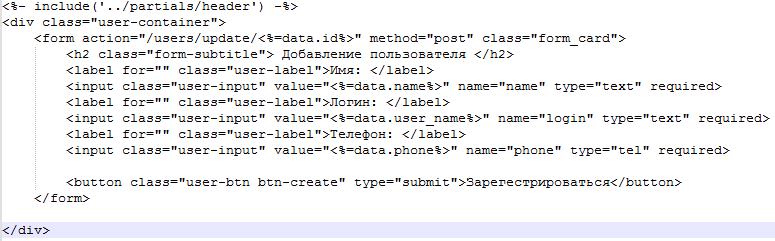


Рисунок 25 – Код добавления пользователя

# Тестирование программного продукта

Одним из этапов разработки любого программного продукта является тестирование и отладка.

Тестирование программного продукта – это процесс проверки работоспособности и качества программного обеспечения, включающий выполнение различных тестов для обнаружения ошибок, уязвимостей и проблем в функционировании программы.

Цель тестирования – обеспечить качество продукта и удовлетворить требования пользователя.

Тестирование программного продукта было разбито на этапы:

1. Отладка
2. Составление тестовых сценариев.
3. Составление чек-листов.
4. Модульное тестирование.
5. Тестирование базы данных методом CRUD.

Отладка – это процесс поиска и исправления ошибок или неполадок в исходном коде какого-либо программного обеспечения.

Отладка проводилась в процессе написания программного кода.

Чек-лист – это основанный на принципах «защиты от ошибок» инструмент управления, который представляет собой лист со списком необходимых проверок и отметками об их выполнении.

В таблице 11 представлен чек-лист успешной авторизации пользователя на сайте.

Таблица 11 – Чек-лист успешной авторизации пользователя

|  |  |
| --- | --- |
| Поле | Описание |
| Тестовый пример# | 1 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Авторизация пользователя |

Продолжение таблицы 11.

|  |  |
| --- | --- |
| Краткое изложение теста | Проводя данный тест, пользователь должен ввести действительные данные: логин и пароль для успешной авторизации. |
| Этапы теста | 1. Пользователь открыл веб-приложение, перешёл на страницу с авторизацией по соответствующей кнопке 2. Пользователь ввел логин и пароль. 3. Нажал на кнопку «авторизоваться». 4. Переход на главную страницу в веб-приложении. |
| Тестовые данные | rocket, 123456 |
| Ожидаемый результат | Успешная авторизация. |
| Статус | Зачёт |
| Предварительное условие | Пользователь должен быть зарегистрирован для прохождения авторизации и помнить корректные логин и пароль. |
| Постусловие | После успешной авторизации, система перенаправляет пользователя на главную страницу. |

В таблице 12 представлен чек-лист, когда пользователь указывает логин, пароль либо роль не верно.

Таблица 12 – Неуспешная авторизация пользователя

|  |  |
| --- | --- |
| Поле | Описание |
| Тестовый пример# | 2 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Авторизация пользователя |
| Краткое изложение теста | Проводя данный тест, пользователь должен ввести неверный логин и пароль, чтобы проверить систему на вывод ошибки. |
| Этапы теста | 1. Пользователь открыл веб-приложение, перешёл на страницу с авторизацией по соответствующей кнопке 2. Пользователь ввел неверные логин и пароль. 3. Нажал на кнопку «Войти». 4. Вывод ошибки и повторная авторизация. |

Продолжение таблицы 12.

|  |  |
| --- | --- |
| Поле | Описание |
| Тестовые данные | rocket, 6543211 |
| Ожидаемый результат | Не успешная авторизация. |
| Статус | Зачёт |
| Предварительное условие | Пользователь должен быть зарегистрирован. |
| Постусловие | После не успешной авторизации, система выдаст ошибку **«**длина символов не должна превышать 6 символов**».** |

В таблице 13 представлен чек-лист для добавления администратором сеанса.

Таблица 13 – Чек-лист добавление сеанса

|  |  |
| --- | --- |
| Поле | Описание |
| Тестовый пример# | 3 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Добавление нового сеанса в планетарий |
| Краткое изложение теста | Проводя данный тест, администратор должен успешно добавить новый сеанс. |
| Этапы теста | 1. Администратор авторизуется и переходит на главную страницу. 2. Нажимает на кнопку добавления нового сеанса. 3. Переходит на страницу добавления сеанса через кнопку «добавить сеанс» 4. Заполняет форму добавления сеанса и подтверждает действие. 5. Сеанс успешно добавляется. |
| Тестовые данные | Сеанс «Юнона», время начала «11:30», время окончания «13:00», дата проведения «6 февраля 2024 года». |
| Ожидаемый результат | Успешное добавление сеанса. |

Продолжение таблицы 13.

|  |  |
| --- | --- |
| Статус | Зачет |
| Предварительное условие | Необходимо заполнить такие поля, как название сеанса, время начала и окончания, а также дату. |
| Постусловие | После успешного добавления сеанса, он успешно будет показан на главной странице. |

В таблице 14 представлен чек-лист неуспешного добавления сеанса.

Таблица 14 – Не успешное добавление лекции

|  |  |
| --- | --- |
| Поле | Описание |
| Тестовый пример# | 4 |
| Приоритет тестирования | Средний |
| Заголовок/название теста | Добавление новой лекции. |
| Краткое изложение теста | Проводя данный тест, администратор должен неуспешно добавить новый сеанс. |
| Этапы теста | 1. Администратор авторизуется и переходит на главную страницу. 2. Выбирает пункт «Добавление сеанса». 3. Переходит на страницу добавления сеанса через кнопку «добавить сеанс» 4. Заполняет форму уже действующего сеанса и подтверждает. 5. Сеанс не добавляется, появляется уведомление о том, что сеанс уже зарегистрирован. |
| Тестовые данные | Сеанс «Юнона», время начала «11:30», время окончания «13:00», дата проведения «6 февраля 2024 года». |
| Ожидаемый результат | Сообщение об ошибки. |
| Статус | Зачет |
| Предварительное условие | Данный сеанс не может проходить 2 раза в одно время. |
| Постусловие | После попытки добавить новый сеанс, указав данные уже действующего сеанса, администратор получает сообщение об ошибке. Чтобы добавить сеанс, необходимо указать другие данные. |

В дипломном проекте будут использоваться тестовые сценарии для проверки функционала программного продукта.

Тестовый сценарий – это документ, описывающий последовательность шагов, необходимых для проведения конкретного теста. Он служит стратегическим руководством для выявления критических ошибок в системе и включает в себя стартовые условия, входные данные, ожидаемые действия пользователя и результаты, содержит информацию о предусловиях, действиях, ожидаемых результатах и критериях оценки прохождения теста.

В данном документе описывается тестовый сценарий для пользователя, оформляющего билет – таблица 15.

Таблица 15 – Тестовый сценарий пользователя.

|  |  |
| --- | --- |
| Поле | Описание |
| Дата теста | 19.12.2023 |
| Приоритет тестирования | Средний |
| Заголовок | Добавление сеанса |
| Этапы теста | Пользователь:   1. Заходит в учетную запись 2. Попадает на главную страницу 3. Выбирает сеанс 4. Нажимает кнопку «Оформить билет» |
| Тестовые данные | Логин и пароль пользователя |
| Ожидаемый результат | Пользователь должен автоматически попасть на главную страницу, появится уведомление об успешном оформлении билета. |
| Фактический результат | Пользователь должен автоматически попасть на главную страницу, появится уведомление об успешном оформлении билета. |

В данном дипломном проекте так же проводилось UNIT TESTING.

UNIT TESTING – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Целью данного тестирования является изолировать отдельные части программы и показать, что по отдельности эти части работоспособны.

В таблице 16 представлен пример тестирования методом CRUD, в котором показаны запросы и данные, которые были выведены с помощью этих запросов.

Таблица 16 – Тестирование CRUD

|  |  |
| --- | --- |
| Запрос | Вывод данных |
| INSERT INTO `users` (`id`, `name`, `phone`, `email`, `role\_id`, `password`, `user\_name`) VALUES (NULL, '3', 'Никита Бурлаков', '89249980809', 'levi@2345', '123456', 'levi'); |  |
| SELECT CONCAT\_WS(' ', users.name) AS 'FIO', users.phone FROM `users`; |  |
| UPDATE `users` SET `name` = Даня Кандалов', `login` = rocket' WHERE `users`.`id` = 2 |  |
| DELETE FROM `users` WHERE users.id = 5 | Пользователь с Id = 5 удалён |

В ходе тестирования программного продукта были проведены различные виды тестов, включая юнит-тесты, CRUD, модульное тестирование и составление чек-листов.

В результате успешного тестирования, программный продукт соответствует требованиям и спецификациям, а также удовлетворяет потребности пользователей и готов к дальнейшей эксплуатации.

1. Документирование программного продукта

# 5.1 Руководство пользователя

Веб-приложение «Планетарий» имеет простой и понятный интерфейс, что не предоставит пользователям сложностей в пользовании веб-приложением.

Для дальнейшей работы с веб-приложением пользователю будет необходимо авторизоваться, либо же, если нет аккаунта, создать его. Страница авторизации представлена на рисунке 25.

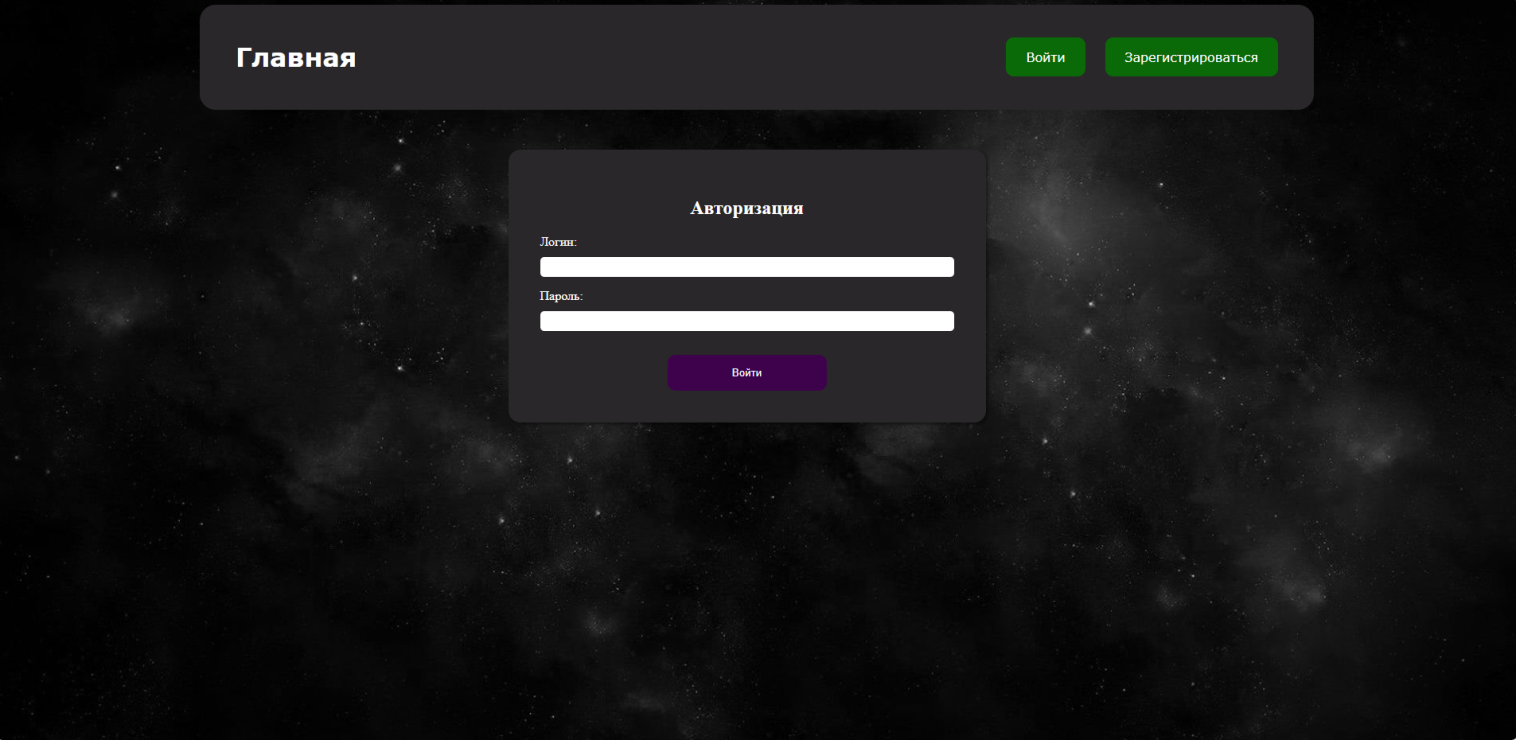


Рисунок 25 – Авторизация

Если аккаунта у пользователя нет, ему необходимо будет создать новый аккаунт. Страница регистрации представлена на рисунке 26.

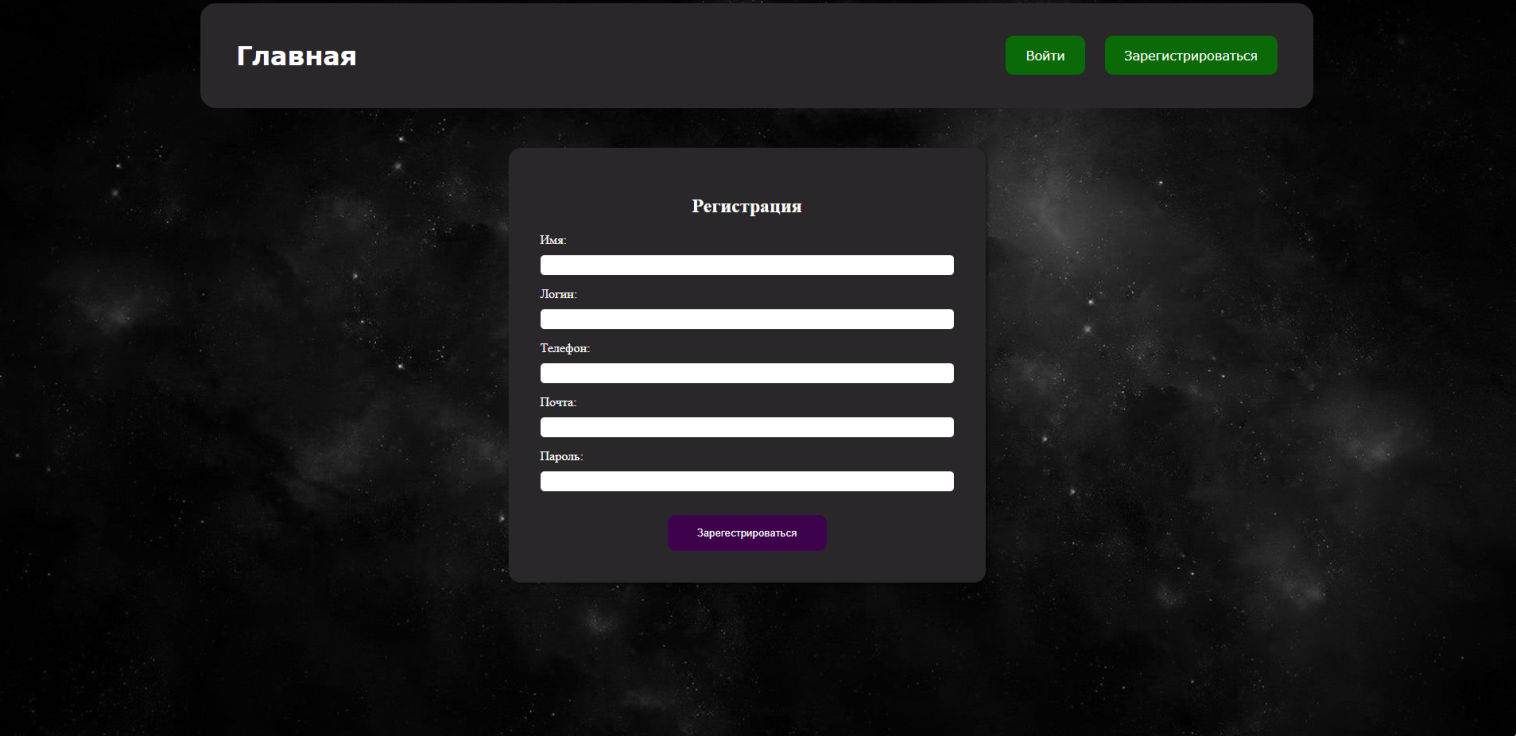


Рисунок 26 – Регистрация

При успешной авторизации пользователь попадает на главную страницу, на которой пользователю предоставляются списки всех сеансов (рисунок 27).

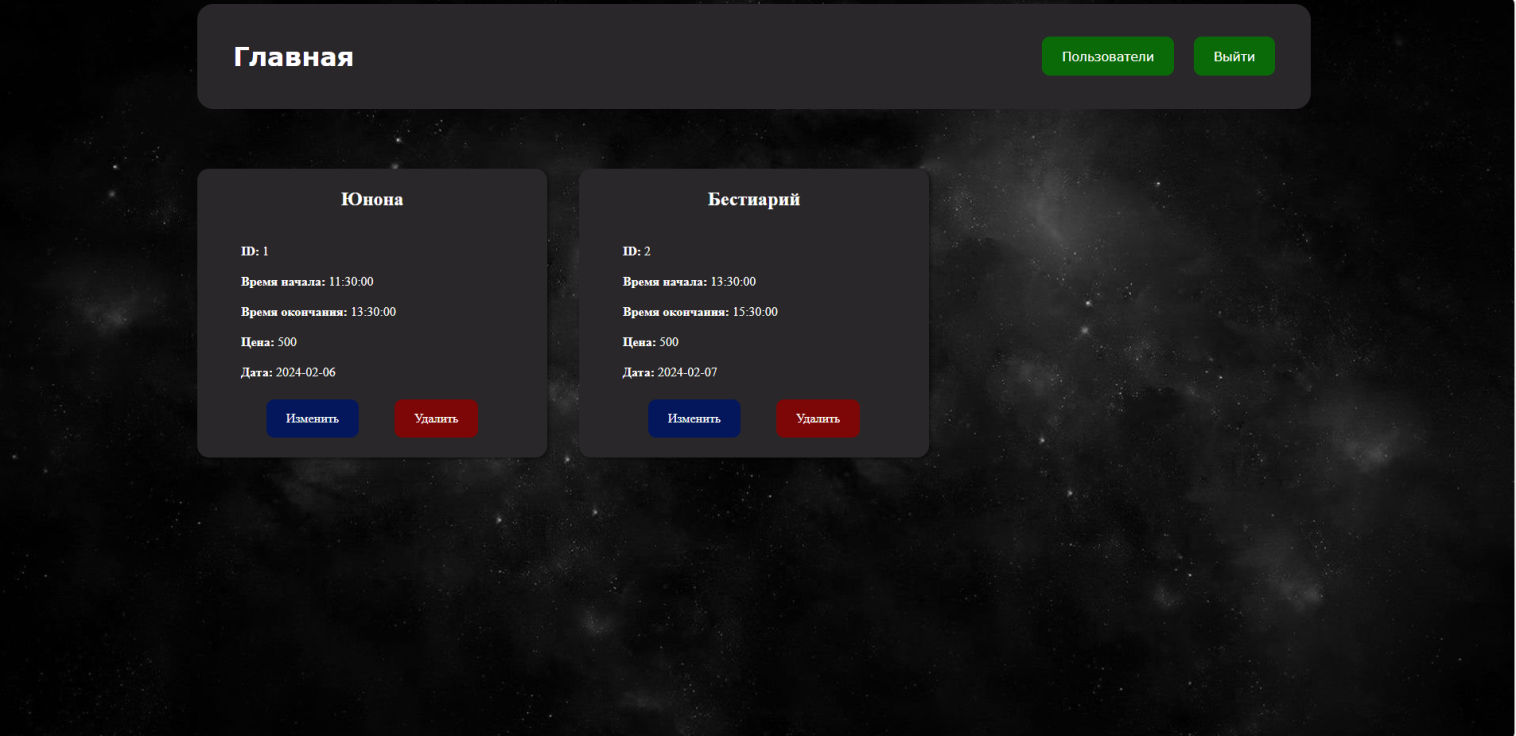


Рисунок 27 – Страница «Главная страница»

Обладая соответствующими правами, то есть правами администратора, предоставляется возможность добавления нового сеанса (рисунок 28).

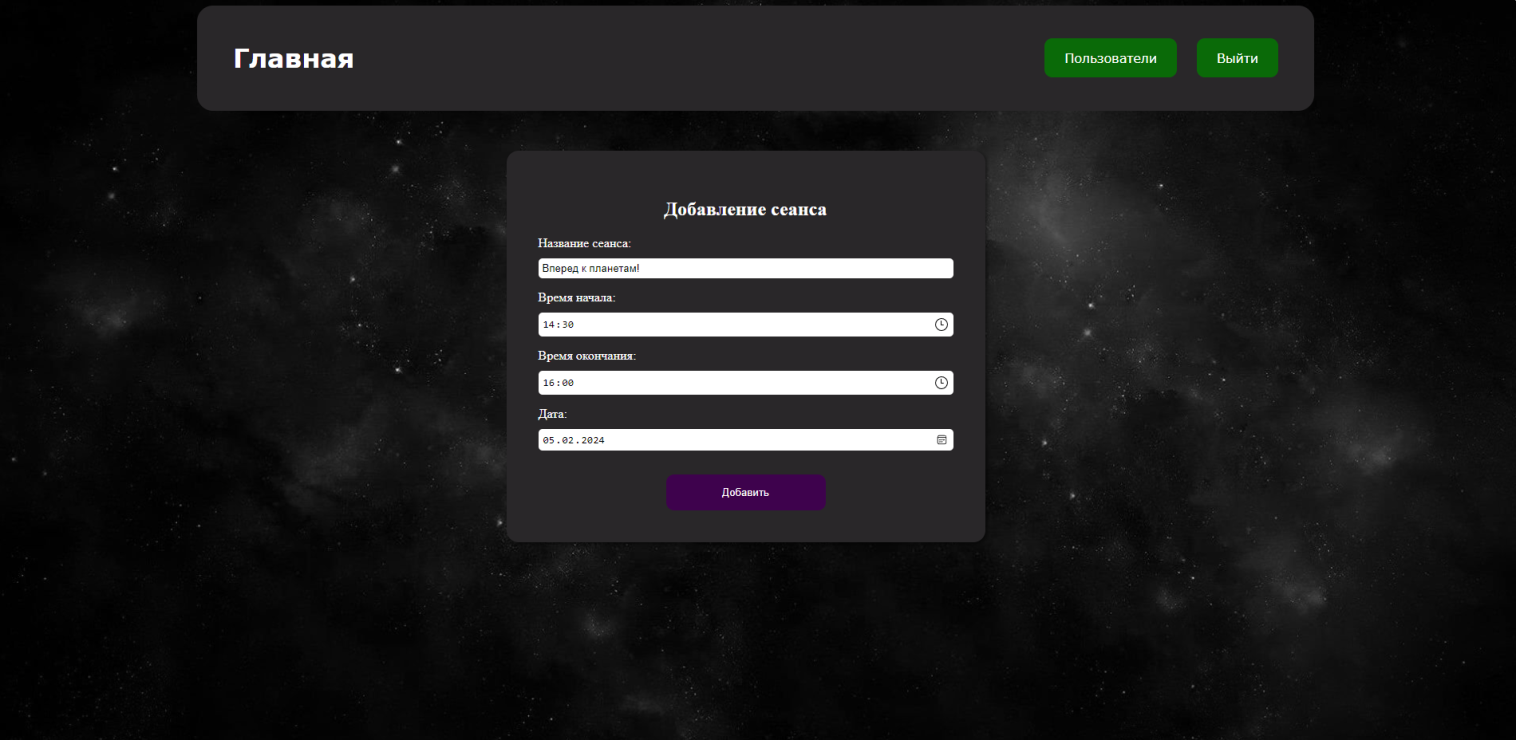


Рисунок 28 – Страница «Регистрации»

Также администратор может редактировать информацию, касающуюся сеанса, а именно: название сеанса, время начала и окончания и дату проведения (рисунок 29).

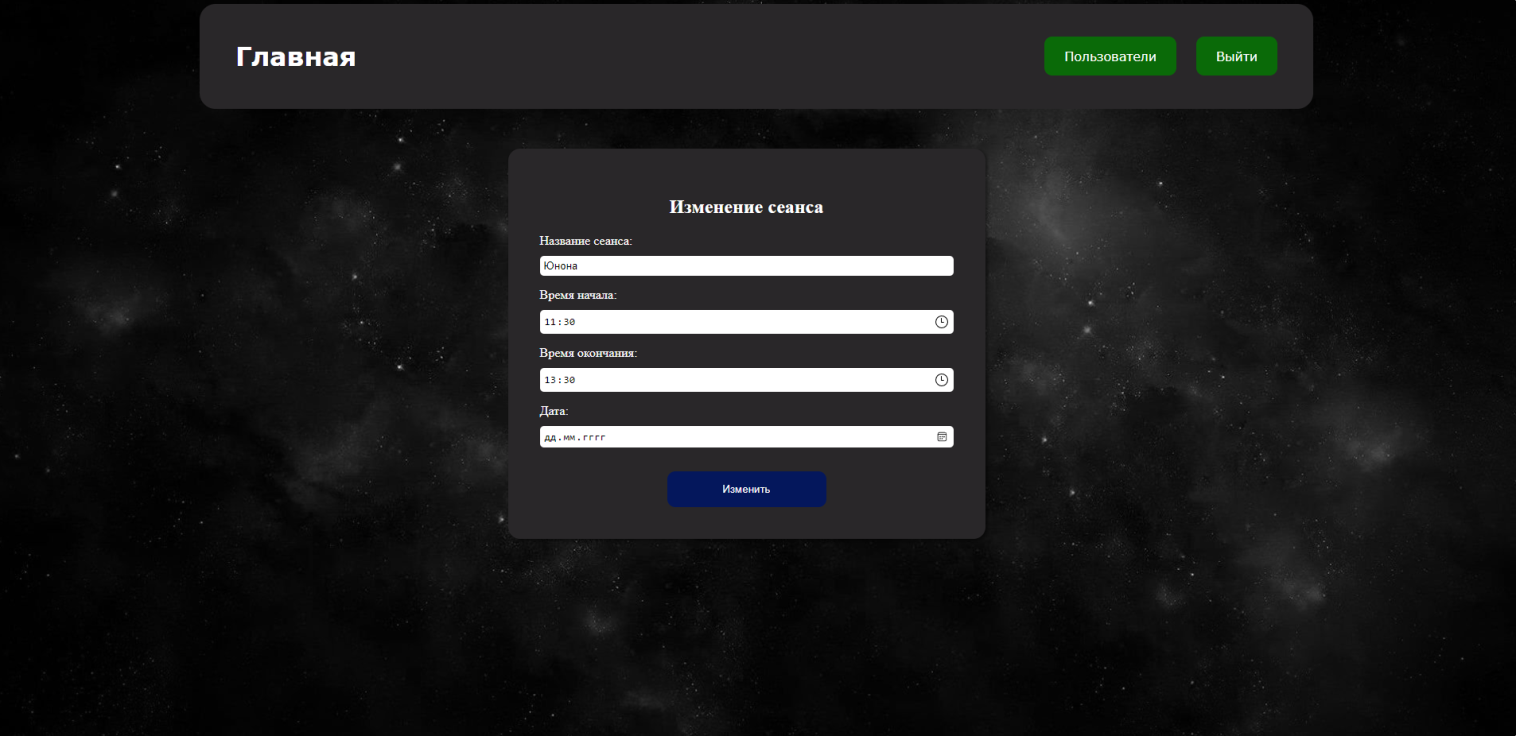


Рисунок 29 – Страница «Изменение сеанса»

# ЗАКЛЮЧЕНИЕ

В ходе работы была поставлена цель разработать веб-приложение «Планетарий». Для достижения поставленной цели, были выполнены следующие задачи:

* Проведено исследование предметной области.
* Проведен анализ инструментальных средств разработки.
* Разработано техническое задание.
* Спроектирован и разработан программный продукт с базой данных.
* Проведено тестирование и отладка.
* Разработано руководство пользователя.

Было проведено исследование и разработана концепция веб-приложения, предназначенного для изучения звездного неба и астрономических явлений. В ходе работы были изучены теоретические основы разработки веб-приложения, выбраны подходящие технологии и инструменты для создания «Планетария».

Были определены требования для разработки данного приложения. Исходя из требований были рассмотрены возможные реализации приложения на различных языках программирования. Выбор был остановлен на языке JS. В разработанном приложении хорошо сочетается интерфейс веб-приложения его функциональностью и простотой использования. Разработаны ER-модель предметной области и шаблоны построения веб-приложения, которые были реализованы на практике. Разработано удобное и доступное для пользователя веб-приложение, имеющее понятные диалоговые окна интерфейса.

Создание веб-приложения «Планетарий» является актуальным и перспективным направлением развития информационных технологий. Оно предоставляет возможность пользователям изучать звездное небо, наблюдать за движением небесных тел и получать информацию о различных астрономических явлениях.

Таким образом, веб-приложение «Планетарий» является успешным примером реализации образовательного проекта для изучения астрономии, который способствует ее популяризации, расширению кругозора и повышению кругозора пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* + - 1. beonmax.com – Курс PHP/MySQL – URL: https://beonmax.com/courses/php-and-mysql/ (дата обращения: 10.03.2024). – Текст: электронный.
      2. bigenc.ru – Солнечная система – URL: https://bigenc.ru/c/solnechnaia-sistema-c236f6 (дата обращения: 26.04.2024). – Текст: электронный.
      3. blog.skillfactory.ru – JavaScript – URL: https://blog.skillfactory.ru/glossary/javascript/ (дата обращения: 02.03.2024). – Текст: электронный.
      4. developer.mozilla.org – Работа с JSON – URL https://developer.mozilla.org/ru/docs/Learn/JavaScript/Objects/JSON (дата обращения: 10.02.2024). – Текст: электронный.
      5. dic.academic.ru – ПЛАНЕТАРИЙ – URL: https://dic.academic.ru/dic.nsf/dic\_fwords/ПЛАНЕТАРИЙ (дата обращения: 11.02.2024). – Текст: электронный.
      6. externat.foxford.ru – Солнечная система: строение и характеристика – URL: https://externat.foxford.ru/polezno-znat/wiki-astronomiya-solnechnaya-sistema (дата обращения: 26.03.2024). – Текст: электронный.
      7. github.com – github.com. – URL: https://github.com/django-ckeditor/django-ckeditor/blob/master/README.rst#required (дата обращения: 11.02.2024). – Текст: электронный.
      8. habr.com – Что такое JSON – URL: https://habr.com/ru/articles/554274/ (дата обращения: 22.11.2023). – Текст: электронный.
      9. html5book.ru – 3.1 Основы JavaScript – URL: https://html5book.ru/osnovy-javascript/ (дата обращения: 15.02.2024). – Текст: электронный.
      10. html5book.ru – HTML-элементы – https://html5book.ru/html-tags/ (дата обращения: 18.03.2024). – Текст: электронный.
      11. kipmu.ru – Созвездия: список, описание, что такое, названия, карта, история, фото и видео – URL: https://kipmu.ru/sozvezdiya/ (дата обращения: 22.02.2024). – Текст: электронный.
      12. kurtaslov.ru – kurtaslov.ru – URL: https://kartaslov.ru/значение-слова/планетарий (дата обращения: 26.03.2024). – Текст: электронный.
      13. lear.javascript.ru – Современный учебник JavaScript. – URL: https://learn.javascript.ru/ (дата обращения: 25.03.2024). – Текст: электронный.
      14. open-source-peace.github.io – JavaScript. Современный учебник – URL: https://open-source-peace.github.io/w3schoolsrus/jslearn/index.html#gsc.tab=0 (дата обращения: 21.03.2024). – Текст: электронный.
      15. servergate.ru – MySQL. Преимущества и недостатки – URL: https://servergate.ru/articles/mysql-preimushchestva-i-nedostatki/ (дата обращения: 01.04.2024). – Текст: электронный.
      16. skillbox.ru – Руководство по NodeJS для новичков – URL: https://skillbox.ru/media/code/chem\_khorosh\_node\_js/ (дата обращения: 01.03.2024). – Текст: электронный.
      17. thecode.media – Что такое JSON – URL: https://thecode.media/json/?ysclid=ls9qt92i1656834121 (дата обращения: 03.04.2024)
      18. timeweb.cloud – Как добавить JavaScript в HTML: инструкция https://timeweb.cloud/tutorials/javascript/kak-dobavit-javascript-v-html (дата обращения: 07.03.2024). – Текст: электронный.
      19. website-create.ru – Вывод информации из базы данных MySql на веб страницу. – URL: https://website-create.ru/vivod-iz-bazi-mysql/?ysclid=lnzjt7pev5677217040 (дата обращения: 12.03.2024). – Текст: электронный.
      20. wiki.rakovets.by – Основы JavaScript – URL: https://wiki.rakovets.by/web/js/ (дата обращения: 15.02.2024). – Текст: электронный.
      21. www.nic.ru – Как встроить JavaScript в HTML – URL: https://www.nic.ru/help/kak-vstroit6-javascript-v-html\_11771.html (дата обращения: 07.03.2024). – Текст: электронный.

Приложение Б – Листинг главной страницы

<%- include('./partials/header') -%>

<div class="create-container">

<a class="header-btn btn-create" href="/session/create">Добавить</a>

<%if(auth.role\_id !== 2 && auth.id) {%>

<form action="/" method="post">

<select name="select" id="select" class="btn-change btn-update" style="border: none;" required>

<option value="1" <%if (selected === '1') { %> selected <% } %>>Все</option>

<option value="2" <%if (selected === '2') { %> selected <%}%>> Мои</option>

</select>

<button type="submit" style="padding: 15px 10px; border-radius: 10px; color: white; font-size: 15px; font-weight: 600; border: none; background-color: rgb(0, 0, 0, 0.); ">Найти</button>

</form>

<%}%>

</div>

<%if(message.length != 0){%>

<div class="message" style="padding: 20px; background: red; height: 200px; min-width: 800px;">

<%=message%>

</div>

<%}%>

<div class="user-cards">

<% data.forEach(function (item){ %>

<div class="user-card">

<h2 class="subtitle"><%=item.name%></h2>

<ul class="user-content">

<li class="user-desc"><b>ID:</b> <%= item.id %></li>

<li class="user-desc"><b>Время начала:</b> <%= item.time\_start %></li>

<li class="user-desc"><b>Время окончания:</b> <%= item.time\_end %></li>

<li class="user-desc"><b>Цена:</b> 500</li>

<li class="user-desc" style="margin-bottom: 25px;"><b>Дата:</b> <%= item.date.toISOString().replace('T', ' ').substring(0, 10) %></li>

</ul>

<% if (auth.id && auth.role\_id === 2) {%>

<div class="card-btn">

<a class="btn-change btn-update" href="/session/update/<%= item.id %>">Изменить</a>

<a class="btn-change btn-delete" href="/session/delete/<%= item.id %>">Удалить</a>

</div>

<% } %>

<% if (auth.id && auth.role\_id === 1) {%>

<div class="card-btn">

<a class="btn-change btn-update" href="/session/buy/<%= item.id%>" style="margin-right: 10;">Купить билет</a>

<a class="btn-change btn-delete" href="/session/unbuy/<%= item.id%>">Отменить запись</a>

</div>

<% } %>

</div>

<% }); %>

</div>

<%- include('./partials/footer') -%>