**Частное образовательное учреждение высшего образования**

**«Международный Институт Дизайна и Сервиса» (ЧОУВО МИДиС)**

Кафедра математики и информатики

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Специальность: 09.02.07 Информационные системы и программирование Квалификация: программист

Форма обучения: очная

Выполнил: Кузнецов В. А.

Группа: П-43с

Проверил: Кондаков С. А.

Челябинск 2025

**Содержание**

[Введение 3](#_Toc190851804)

[Глава 1. База практики 5](#_Toc190851805)

[Выводы по Главе 1 7](#_Toc190851806)

[Глава 2. Проектирование и разработка информационных систем 8](#_Toc190851807)

[2.1 Анализ предметной области, выбор средств разработки 8](#_Toc190851808)

[2.2 Проектирование и разработка информационной системы для контроля выполнения нагрузки преподавателей ВУЗа. 9](#_Toc190851809)

[2.3 Проектирование и разработка информационной системы для работника склада (вариант: склад торговой организации, занимающейся сбытом как продукции собственного производства, так и продукции внешних поставщиков;). 13](#_Toc190851810)

[2.4 Проектирование и разработка информационной системы для детского сада 16](#_Toc190851811)

[2.5 Проектирование и разработка информационной системы для управления тренажерным залом свободного доступа 21](#_Toc190851812)

[2.6 Проектирование и разработка информационной системы для домашней библиотеки 24](#_Toc190851813)

[2.7 Проектирование и разработка информационной системы для контроля сессионной успеваемости студентов ВУЗа 28](#_Toc190851814)

[2.8 Проектирование и разработка информационной системы для организации дипломного проектирования в ВУЗе 32](#_Toc190851815)

[2.9 Проектирование и разработка информационной системы для студенческого профкома ВУЗа 36](#_Toc190851816)

[2.10 Проектирование и разработка информационной системы для работника склада (вариант: склад оптовой торговли организации) 41](#_Toc190851817)

[2.11 Проектирование и разработка информационной системы для библиотеки ВУЗа 45](#_Toc190851818)

[Выводы по Главе 1 48](#_Toc190851819)

[Глава 3. Создание веб-приложения для изучения физики 50](#_Toc190851820)

[3.1 Техническое задание, выбор средств разработки 50](#_Toc190851821)

[3.2 Разработка прототипа приложения 50](#_Toc190851822)

[3.3 Анализ цветовой политики МИДиС, разработка дизайна приветственной страницы приложения. 51](#_Toc190851823)

[3.4 Разработка базы данных 54](#_Toc190851824)

[Приложение 57](#_Toc190851825)

# Введение

Производственная практика направлена на формирование у обучающихся практических профессиональных умений, приобретение первоначального практического опыта, реализуется в рамках профессиональных модулей образовательной программы по основным видам профессиональной деятельности для последующего освоения ими общих и профессиональных компетенций по специальности 09.02.07 Информационные системы и программирование.

Производственная практика организуется в форме практической подготовки и предусмотрена календарным учебным графиком в течении 9 недель в рамках профессиональных модулей по специальности 09.02.07 Информационные системы и программирование.

Главными целями производственной практики являются:

Комплексное освоение обучающимися всех видов профессиональной деятельности по специальности 09.02.07 Информационные системы и программирование;

Формирование общих и профессиональных компетенций, а также приобретение необходимых умений и опыта практической работы по специальности 09.02.07 Информационные системы и программирование;

Повышение качества использования в практической деятельности новых знаний и умений, стремления к саморазвитию;

Осознание социальной значимости своей будущей профессии и мотивации к выполнению профессиональной деятельности.

Задачи производственной практики:

Знакомство с основами будущей профессиональной деятельности;

Изучение современных информационных технологий получения и обработки данных;

Приобретение обучающими практического опыта, знаний и умений для самостоятельной разработки веб-сайтов;

Развитие практических навыков по работе в профессиональной среде разработки Microsoft Visual Studio Code;

Выработка навыков создания обзоров научной литературы и электронных информационно-образовательных ресурсов для профессиональной деятельности;

Знакомство с рабочей атмосферой;

Закрепление навыков вёрстки;

Рабочая программа производственной практики является частью образовательной программы по специальности 09.02.07 Информационные системы и программирование в части освоения основных видов профессиональной деятельности:

Проектирование и разработка информационных систем;

Разработка дизайна веб-приложений;

Проектирование, разработка и оптимизация веб-приложений.

# Глава 1. База практики

Практика проходила в Международном институте дизайна и сервиса

«МИДиС»[12] . Международный институт дизайна и сервиса – это комплекс учебных заведений, включающий школу, колледж и институт. Колледж при МИСиС предлагает обучение по различным специальностям, в том числе:

* Информационные системы и Программирование: Обучение охватывает разработку и управление информационными системами, а также программирование, включая основные языки программирования и принципы разработки программного обеспечения.
* Коммерция: Студенты изучают основы бизнеса, менеджмента, маркетинга и экономики, чтобы приобрести знания и навыки, необходимые для успешной карьеры в сфере коммерции.
* Бизнес-информатика: Эта область позволяет студентам получить знания и навыки в области проектирования и поддержки архитектуры предприятия, включая системы целей, бизнес-процессы, организационную структуру и информационные технологии на этапе цифровой трансформации.
* Лингвистика: Бизнес-лингвистика предполагает изучение функционирования языка в бизнесе и лингвистических аспектов делового общения.
* Прикладная информатика: Сочетает в себе компьютерные науки, экономику и менеджмент. Она фокусируется на информационных технологиях для бизнеса. Основная цель программы «бизнес-информатика» – подготовить менеджеров, способных решать бизнес-задачи и управлять бизнесом с помощью информационных технологий[12].

Институт оснащен инструментами и оборудованием для выполнения практических заданий по веб-разработке и программированию, которые будут необходимы для достижения наших целей.

Для выполнения поставленных задач нам понадобится следующее:

Visual Studio Code[5]: Это бесплатный и легкий текстовый редактор, разработанный компанией Microsoft. Он предназначен для разработки различных типов приложений, включая веб-приложения, мобильные приложения и облачные приложения.

Обладая таким широким набором инструментов и опытом разработки программного обеспечения, мы сможем выполнить поставленные задачи и приобрести необходимые компетенции.

Кабинеты для практики оборудованы компьютерами, стульями и столами для студентов. В каждом кабинете также есть проекторы и проекционные экраны, а также маркерные доски для преподавателей.

# Выводы по Главе 1

В этой главе была представлена база практики в Международном институте дизайна и сервиса (МИДИС), предлагающем обучение по различным специальностям. Оснащенные необходимыми инструментами и программным обеспечением, таким как Visual Studio Code[5], мы хорошо подготовлены к выполнению практических заданий по веб-разработке и программированию. Практические кабинеты полностью оборудованы, что обеспечивает благоприятную среду для обучения. Эта глава закладывает основу для нашей стажировки, где мы получим необходимые навыки и знания.

# Глава 2. Проектирование и разработка информационных систем

## 2.1 Анализ предметной области, выбор средств разработки

Перед началом разработки информационных систем было проведено тщательное изучение и анализ предметной области. Основной задачей являлось создание 10 информационных систем, каждая из которых должна была включать функционал входа и регистрации пользователей, а также взаимодействие с базами данных. Данные системы предназначены для автоматизации процессов, связанных с обработкой и хранением информации, что требует обеспечения удобного интерфейса, надежности и масштабируемости.

В рамках анализа предметной области были определены ключевые требования к разрабатываемым системам:

* Интуитивно понятный интерфейс: Системы должны быть доступны для пользователей с разным уровнем технической подготовки.
* Кроссплатформенность: Возможность работы систем на различных устройствах и операционных системах.
* Простота разработки и поддержки: Выбор технологий, которые позволяют быстро разрабатывать и в дальнейшем поддерживать системы.

На основании этих требований были выбраны следующие средства разработки: HTML, CSS и JavaScript.

**Обоснование выбора технологий**

**HTML (HyperText Markup Language):**

* Является стандартным языком разметки для создания веб-страниц. HTML позволяет структурировать контент и обеспечивает базовую организацию элементов интерфейса.
* Выбор HTML обусловлен его универсальностью и поддержкой всеми современными браузерами, что обеспечивает кроссплатформенность разрабатываемых систем.

**CSS (Cascading Style Sheets):**

* Используется для оформления и стилизации веб-страниц. CSS позволяет создавать адаптивные и визуально привлекательные интерфейсы, что соответствует требованию интуитивно понятного интерфейса.
* С помощью CSS можно легко адаптировать интерфейс под различные устройства, что особенно важно в условиях разнообразия пользовательских устройств.

**JavaScript:**

* Является основным языком программирования для создания интерактивных элементов на стороне клиента. JavaScript позволяет реализовать динамическое поведение страниц, включая обработку форм, валидацию данных и взаимодействие с сервером без перезагрузки страницы.
* Использование JavaScript также позволяет интегрировать системы с базами данных через API, что обеспечивает функционал регистрации и авторизации пользователей.

## 2.2 Проектирование и разработка информационной системы для контроля выполнения нагрузки преподавателей ВУЗа.

Сначала я спроектировал базу данных для данного сервиса. Перед началом задания, связанном с формированием отношений между актерами и прецендентами, я провел анализ сущностей, и установление связей между ними, выбрал необходимые типы данных для каждого из полей сущности и вывел результат в финальную диаграмму (см. рис. 1)

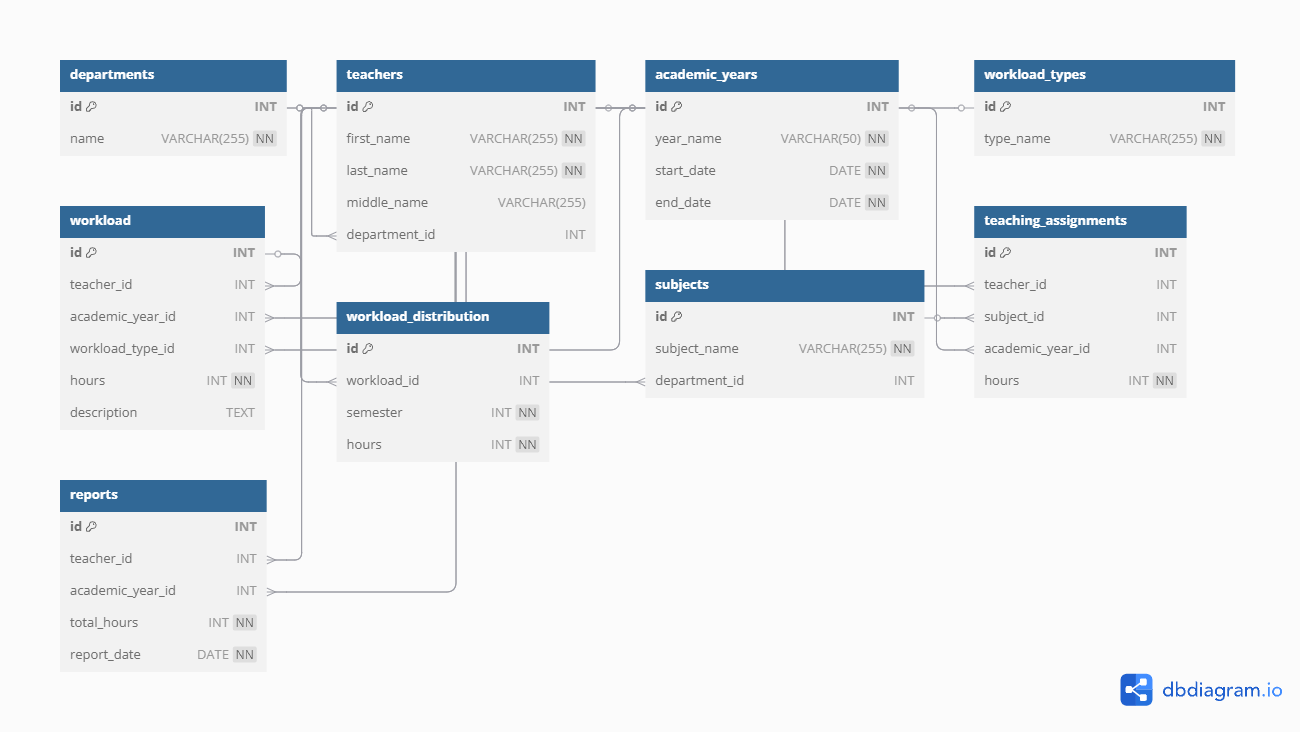


Рис. 1 - База данных для информационной системы для контроля выполнения нагрузки преподавателей ВУЗа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

Актеры:

Преподаватель:

* Основной пользователь системы, который взаимодействует с ней для управления своей учебной нагрузкой.

Администратор:

* Ответственный за управление данными системы (добавление преподавателей, курсов, типов нагрузки и т.д.).

Деканат/Руководство:

* Пользователь, который просматривает отчеты и утверждает выполненную нагрузку.

Система учета нагрузки:

* Внешняя система, которая может интегрироваться с текущей системой для обмена данными.

Прецеденты:

Управление профилем преподавателя:

* Преподаватель может просматривать и редактировать свои личные данные.
* Администратор может добавлять, редактировать или удалять данные преподавателей.

Управление учебной нагрузкой:

* Преподаватель может просматривать свою учебную нагрузку.
* Преподаватель может добавлять или редактировать данные о выполненной нагрузке.
* Администратор может назначать нагрузку преподавателям.

Управление курсами:

* Администратор может добавлять, редактировать или удалять курсы.
* Преподаватель может просматривать список курсов, которые он ведет.

Формирование отчетов:

* Преподаватель может формировать отчет о выполненной нагрузке.
* Деканат/Руководство может просматривать и утверждать отчеты.

Просмотр статистики:

* Деканат/Руководство может просматривать статистику по выполненной нагрузке всех преподавателей.
* Преподаватель может просматривать свою статистику.

Интеграция с внешней системой:

* Система учета нагрузки может обмениваться данными с текущей системой (например, импорт/экспорт данных).

Связи между актерами и прецедентами:

Преподаватель:

* Управление профилем преподавателя.
* Управление учебной нагрузкой.
* Формирование отчетов.
* Просмотр статистики.
* Управление курсами (просмотр).

Администратор:

* Управление профилем преподавателя.
* Управление учебной нагрузкой.
* Управление курсами.

Деканат/Руководство:

* Формирование отчетов (утверждение).
* Просмотр статистики.

Система учета нагрузки:

* Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 2).

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2 – Дизайн приветственной страницы приложения информационной системы для контроля выполнения нагрузки преподавателей ВУЗа

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.3 Проектирование и разработка информационной системы для работника склада (вариант: склад торговой организации, занимающейся сбытом как продукции собственного производства, так и продукции внешних поставщиков;).

Сначала я спроектировал базу данных для данного сервиса (см. рис. 3).

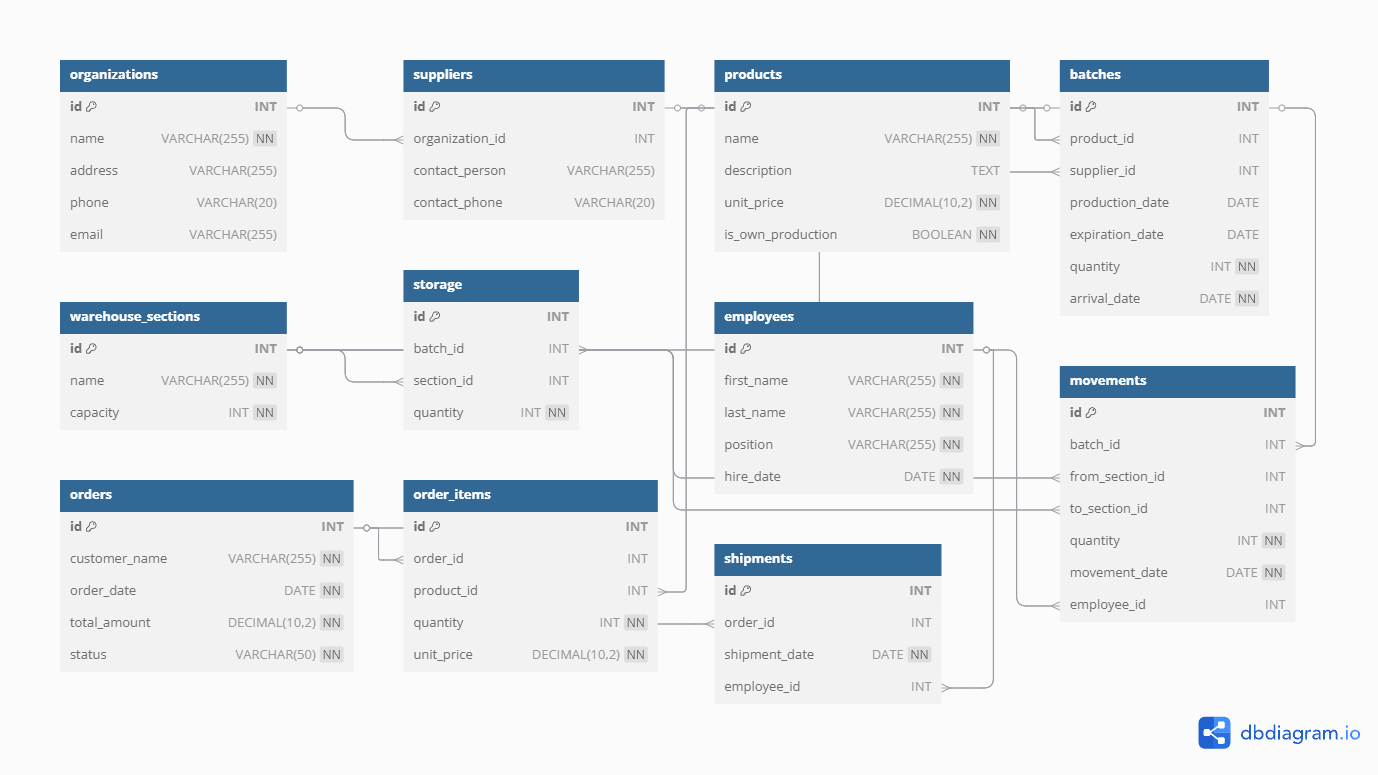


Рис. 3 - База данных для информационной системы для работника склада (вариант: склад торговой организации, занимающейся сбытом как продукции собственного производства, так и продукции внешних поставщиков;)

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

Актеры:

Работник склада:

• Основной пользователь системы, который взаимодействует с ней для управления товарами и заказами.

Администратор:

• Ответственный за управление данными системы (добавление товаров, поставщиков, категорий и т.д.).

Поставщик:

• Внешний участник, предоставляющий товары для склада.

Клиент:

• Покупатель, оформляющий заказы на товары склада.

Прецеденты:

Управление профилем пользователя:

• Работник склада может просматривать и редактировать свои личные данные.

• Администратор может добавлять, редактировать или удалять учетные записи работников.

Управление товарами:

• Работник склада может просматривать список товаров, их количество и категории.

• Администратор может добавлять, редактировать или удалять товары.

• Работник склада может обновлять количество товаров на складе.

Управление заказами:

• Клиент может оформлять заказы на товары.

• Работник склада может обрабатывать заказы, изменять их статус.

• Администратор может просматривать все заказы.

Управление поставщиками:

• Администратор может добавлять, редактировать или удалять поставщиков.

• Работник склада может просматривать список поставщиков и их контактные данные.

Формирование отчетов:

• Администратор может формировать отчеты о движении товаров и заказах.

• Работник склада может просматривать отчеты о запасах.

Просмотр статистики:

• Администратор может просматривать статистику по поставкам, продажам и складским запасам.

• Работник склада может просматривать текущие остатки товаров.

Связи между актерами и прецедентами:

Работник склада:

• Управление профилем.

• Управление товарами.

• Управление заказами.

• Просмотр поставщиков.

• Просмотр отчетов.

• Просмотр статистики.

Администратор:

• Управление профилем работников.

• Управление товарами.

• Управление заказами.

• Управление поставщиками.

• Формирование отчетов.

• Просмотр статистики.

Поставщик:

• Предоставление товаров для склада.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 4).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4 – Дизайн приветственной страницы приложения информационной системы для работника склада (вариант: склад торговой организации, занимающейся сбытом как продукции собственного производства, так и продукции внешних поставщиков;)

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.4 Проектирование и разработка информационной системы для детского сада

Сначала я спроектировал базу данных для данного сервиса (см. рис. 5).

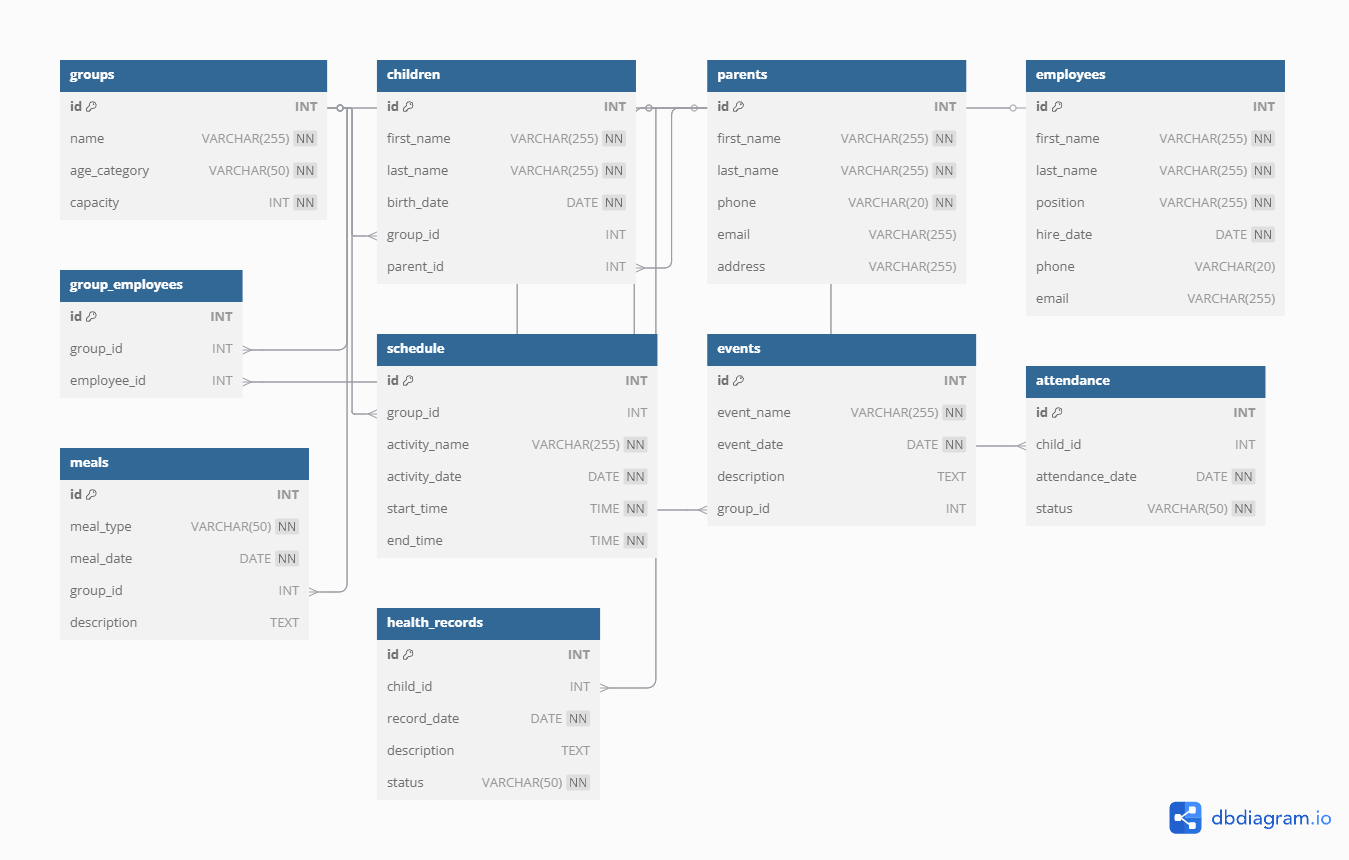


Рис. 5 - База данных для информационной системы детского сада

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Актеры:

Администратор:

* + Ответственный за настройку и управление системой. Администратор может управлять учетными записями пользователей, добавлять данные о сотрудниках и группах, а также формировать отчеты и статистику.

Воспитатель:

* + Основной пользователь системы, который взаимодействует с ней для управления детьми, занятием, посещаемостью и другой образовательной деятельностью. Воспитатель также может отслеживать медицинскую информацию о детях.

Родитель:

* + Внешний пользователь системы, который может отслеживать информацию о своем ребенке, посещаемости, занятиях, а также получать уведомления и сообщения от воспитателей.

Медицинский работник:

* + Пользователь, отвечающий за медицинские данные детей. Может вводить информацию о состоянии здоровья, вакцинациях и аллергиях.

Администратор системы:

* + Человек, занимающийся техническим обслуживанием и настройкой системы, но не напрямую взаимодействующий с данными детей или их родителями. Отвечает за поддержку системы.

Прецеденты:

Управление профилем пользователя:

* + Воспитатель может просматривать и редактировать свою личную информацию (контактные данные, график работы).
  + Родитель может просматривать данные о своем ребенке, получать отчеты и уведомления.
  + Администратор может создавать, редактировать и удалять учетные записи пользователей (воспитателей, родителей, медицинских работников и т.д.).

Управление детьми:

* + Воспитатель может добавлять, редактировать и удалять информацию о детях, распределять их по группам.
  + Администратор может добавлять и редактировать информацию о детях и их группах.

Управление посещаемостью:

* + Воспитатель может фиксировать посещаемость детей (присутствие, отсутствие, опоздание).
  + Родитель может просматривать посещаемость своего ребенка.
  + Администратор может просматривать все данные по посещаемости.

Управление занятиями:

* + Воспитатель может планировать и проводить занятия, фиксировать присутствие детей на занятиях.
  + Родитель может получать уведомления о проведении занятий, а также следить за успехами ребенка.
  + Администратор может просматривать и управлять расписанием занятий.

Медицинская информация:

* + Медицинский работник может добавлять и редактировать медицинскую информацию о детях (аллергии, прививки, заболевания).
  + Воспитатель и родитель могут просматривать медицинские данные ребенка, но только в пределах своей компетенции.
  + Администратор может просматривать и редактировать медицинские данные детей.

Формирование отчетов и статистики:

* + Администратор может формировать отчеты по посещаемости, занятиям, медицинским данным и другим аспектам работы детского сада.
  + Воспитатель может просматривать отчеты по своей группе (например, посещаемость или успеваемость).
  + Родитель может получать отчеты о развитии и занятиях своего ребенка.

Связи между актерами и прецедентами:

Воспитатель:

* + Управление профилем.
  + Управление детьми.
  + Управление посещаемостью.
  + Управление занятиями.
  + Просмотр медицинской информации.
  + Просмотр отчетов и статистики.

Администратор:

* + Управление профилем пользователей.
  + Управление детьми.
  + Управление посещаемостью.
  + Управление занятиями.
  + Управление медицинской информацией.
  + Формирование отчетов и статистики.

Родитель:

* + Просмотр профиля ребенка.
  + Просмотр посещаемости ребенка.
  + Просмотр отчетов о занятиях.
  + Просмотр медицинской информации ребенка.

Медицинский работник:

* + Управление медицинской информацией.
  + Просмотр медицинских данных детей.

Администратор системы:

* + Техническая поддержка системы и управление доступами.  
    После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 6).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 6 – Дизайн приветственной страницы приложения информационной системы для детского сада

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.5 Проектирование и разработка информационной системы для управления тренажерным залом свободного доступа

Сначала я спроектировал базу данных для данного сервиса (см. рис. 7).

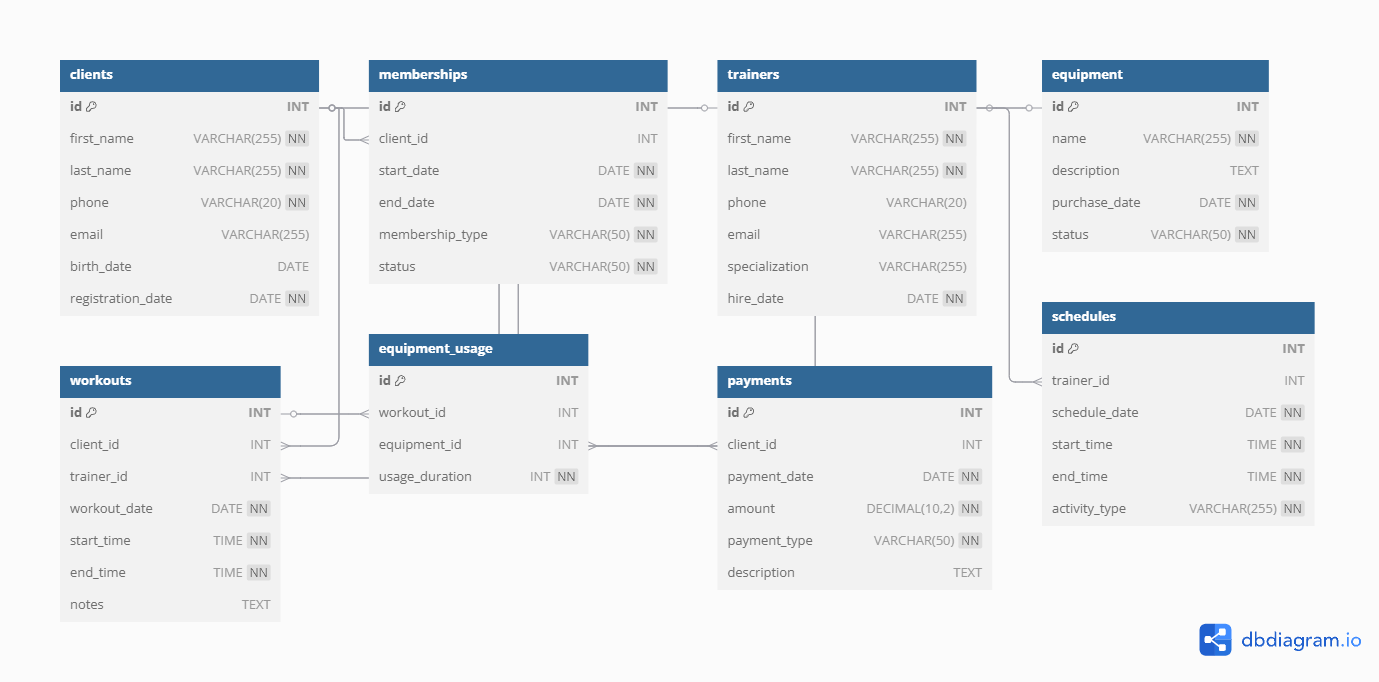


Рис. 7 - База данных для информационной системы управления тренажерным залом свободного доступа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

1. **Клиент:**
   * Основной пользователь системы, который взаимодействует с ней для записи на тренировки, просмотра своего расписания и управления своим абонементом.
2. **Тренер:**
   * Пользователь, который управляет тренировками, просматривает расписание занятий, а также ведет учет посещаемости клиентов.
3. **Администратор:**
   * Ответственный за управление данными системы (добавление клиентов, тренеров, абонементов, расписания и т.д.).
4. **Руководство:**
   * Пользователь, который просматривает отчеты и анализирует статистику по работе тренажерного зала.
5. **Система оплаты:**
   * Внешняя система, которая может интегрироваться с текущей системой для обработки платежей и управления финансовыми данными.

**Прецеденты:**

1. **Управление профилем клиента:**
   * Клиент может просматривать и редактировать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные клиентов.
2. **Управление абонементами:**
   * Клиент может просматривать свои абонементы и приобретать новые.
   * Администратор может добавлять, редактировать или удалять абонементы.
3. **Управление расписанием тренировок:**
   * Тренер может просматривать и редактировать расписание тренировок.
   * Клиент может просматривать расписание и записываться на тренировки.
   * Администратор может управлять расписанием тренировок.
4. **Учет посещаемости:**
   * Тренер может отмечать посещаемость клиентов на тренировках.
   * Клиент может просматривать свою историю посещений.
5. **Формирование отчетов:**
   * Тренер может формировать отчеты по посещаемости и эффективности тренировок.
   * Руководство может просматривать и анализировать отчеты.
6. **Просмотр статистики:**
   * Руководство может просматривать статистику по работе тренажерного зала (посещаемость, доходы и т.д.).
   * Клиент может просматривать свою статистику (количество посещений, прогресс и т.д.).
7. **Интеграция с системой оплаты:**
   * Система оплаты может обмениваться данными с текущей системой (например, обработка платежей за абонементы).

**Связи между актерами и прецедентами:**

1. **Клиент:**
   * Управление профилем клиента.
   * Управление абонементами.
   * Управление расписанием тренировок (просмотр и запись).
   * Учет посещаемости (просмотр истории).
   * Просмотр статистики.
2. **Тренер:**
   * Управление расписанием тренировок.
   * Учет посещаемости.
   * Формирование отчетов.
   * Просмотр статистики.
3. **Администратор:**
   * Управление профилем клиента.
   * Управление абонементами.
   * Управление расписанием тренировок.
4. **Руководство:**
   * Формирование отчетов (анализ).
   * Просмотр статистики.
5. **Система оплаты:**
   * Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 8).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 8 – Дизайн приветственной страницы приложения информационной системы для управления тренажерным залом свободной посещаемости

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.6 Проектирование и разработка информационной системы для домашней библиотеки

Сначала я спроектировал базу данных для данного сервиса (см. рис. 9).

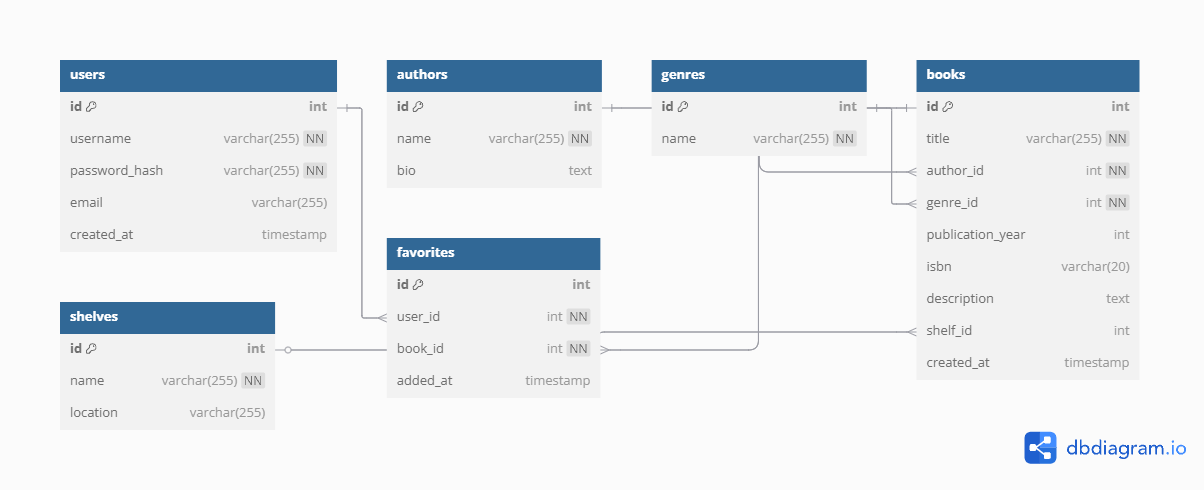


Рис. 9 - База данных для информационной системы домашней библиотеки

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

1. **Пользователь:**
   * Основной пользователь системы, который взаимодействует с ней для добавления книг, поиска, управления своей коллекцией и просмотра статистики.
2. **Администратор:**
   * Ответственный за управление данными системы (добавление, редактирование или удаление книг, авторов, жанров и полок).
3. **Гость:**
   * Пользователь, который может просматривать каталог книг без возможности редактирования или добавления.
4. **Система рекомендаций:**
   * Внешняя система, которая может интегрироваться с текущей системой для предоставления рекомендаций по книгам.

**Прецеденты:**

1. **Управление профилем пользователя:**
   * Пользователь может просматривать и редактировать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные пользователей.
2. **Управление книгами:**
   * Пользователь может добавлять, редактировать или удалять книги из своей коллекции.
   * Администратор может управлять всеми книгами в системе (добавлять, редактировать, удалять).
3. **Управление авторами:**
   * Администратор может добавлять, редактировать или удалять авторов.
4. **Управление жанрами:**
   * Администратор может добавлять, редактировать или удалять жанры.
5. **Управление полками:**
   * Пользователь может создавать и управлять своими полками для организации книг.
   * Администратор может управлять всеми полками в системе.
6. **Поиск книг:**
   * Пользователь и гость могут искать книги по названию, автору, жанру или году издания.
7. **Формирование отчетов:**
   * Пользователь может формировать отчеты о своей коллекции (например, список книг по жанру или автору).
   * Администратор может формировать отчеты по всей библиотеке (например, статистика по жанрам или авторам).
8. **Просмотр статистики:**
   * Пользователь может просматривать статистику своей коллекции (например, количество книг по жанрам).
   * Администратор может просматривать общую статистику библиотеки.
9. **Интеграция с системой рекомендаций:**
   * Система рекомендаций может предоставлять пользователю рекомендации по книгам на основе его предпочтений.

**Связи между актерами и прецедентами:**

1. **Пользователь:**
   * Управление профилем пользователя.
   * Управление книгами (добавление, редактирование, удаление).
   * Управление полками.
   * Поиск книг.
   * Формирование отчетов.
   * Просмотр статистики.
2. **Администратор:**
   * Управление профилем пользователя.
   * Управление книгами.
   * Управление авторами.
   * Управление жанрами.
   * Управление полками.
   * Формирование отчетов.
   * Просмотр статистики.
3. **Гость:**
   * Поиск книг.
4. **Система рекомендаций:**
   * Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 10 – Дизайн приветственной страницы приложения информационной системы для домашней библиотеки

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.7 Проектирование и разработка информационной системы для контроля сессионной успеваемости студентов ВУЗа

Сначала я спроектировал базу данных для данного сервиса. Я провел анализ сущностей, и установление связей между ними, выбрал необходимые типы данных для каждого из полей сущности (см. рис. 11).

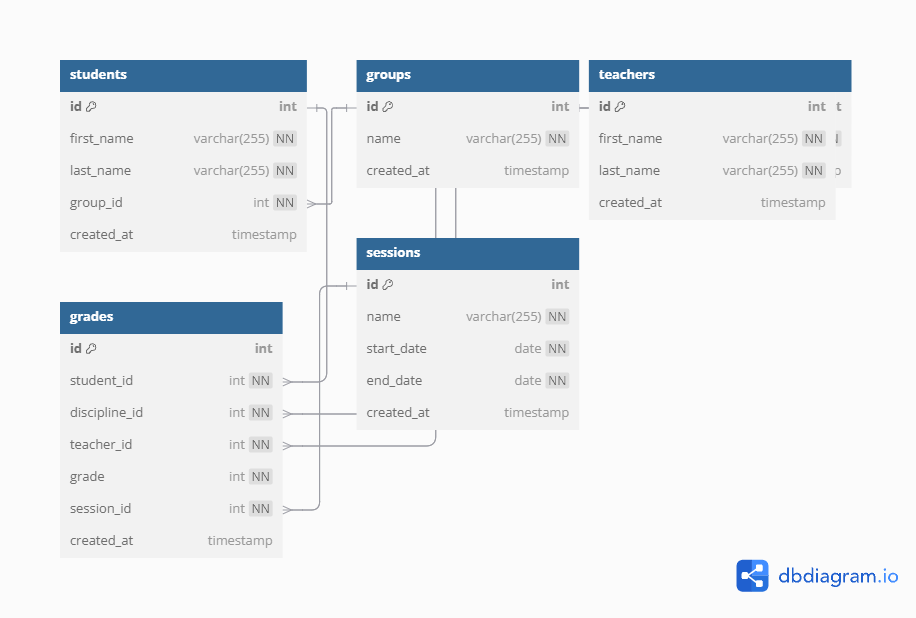


Рис. 11 - База данных для информационной системы для контроля сессионной успеваемости студентов ВУЗа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

Актеры:

1. Студент:
   * Основной пользователь системы, который взаимодействует с ней для просмотра своих оценок и успеваемости.
2. Преподаватель:
   * Пользователь, который выставляет оценки студентам по дисциплинам.
3. Администратор:
   * Ответственный за управление данными системы (добавление, редактирование или удаление студентов, групп, дисциплин, преподавателей и сессий).
4. Деканат:
   * Пользователь, который просматривает отчеты и анализирует успеваемость студентов.
5. Система уведомлений:
   * Внешняя система, которая может интегрироваться с текущей системой для отправки уведомлений студентам и преподавателям.

Прецеденты:

1. Управление профилем студента:
   * Студент может просматривать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные студентов.
2. Управление группами:
   * Администратор может добавлять, редактировать или удалять группы.
3. Управление дисциплинами:
   * Администратор может добавлять, редактировать или удалять дисциплины.
4. Управление преподавателями:
   * Администратор может добавлять, редактировать или удалять данные преподавателей.
5. Выставление оценок:
   * Преподаватель может выставлять оценки студентам по дисциплинам.
   * Студент может просматривать свои оценки.
6. Управление сессиями:
   * Администратор может создавать и управлять сессиями (например, осенняя сессия 2023).
7. Формирование отчетов:
   * Преподаватель может формировать отчеты по успеваемости студентов.
   * Деканат может просматривать и анализировать отчеты по успеваемости.
8. Просмотр статистики:
   * Деканат может просматривать статистику успеваемости студентов (например, средний балл по группам или дисциплинам).
   * Студент может просматривать свою статистику (например, средний балл за сессию).
9. Интеграция с системой уведомлений:
   * Система уведомлений может отправлять уведомления студентам и преподавателям (например, о новых оценках или начале сессии).

Связи между актерами и прецедентами:

1. Студент:
   * Управление профилем студента (просмотр).
   * Просмотр оценок.
   * Просмотр статистики.
2. Преподаватель:
   * Выставление оценок.
   * Формирование отчетов.
3. Администратор:
   * Управление профилем студента (добавление, редактирование, удаление).
   * Управление группами.
   * Управление дисциплинами.
   * Управление преподавателями.
   * Управление сессиями.
4. Деканат:
   * Формирование отчетов (анализ).
   * Просмотр статистики.
5. Система уведомлений:
   * Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 12).

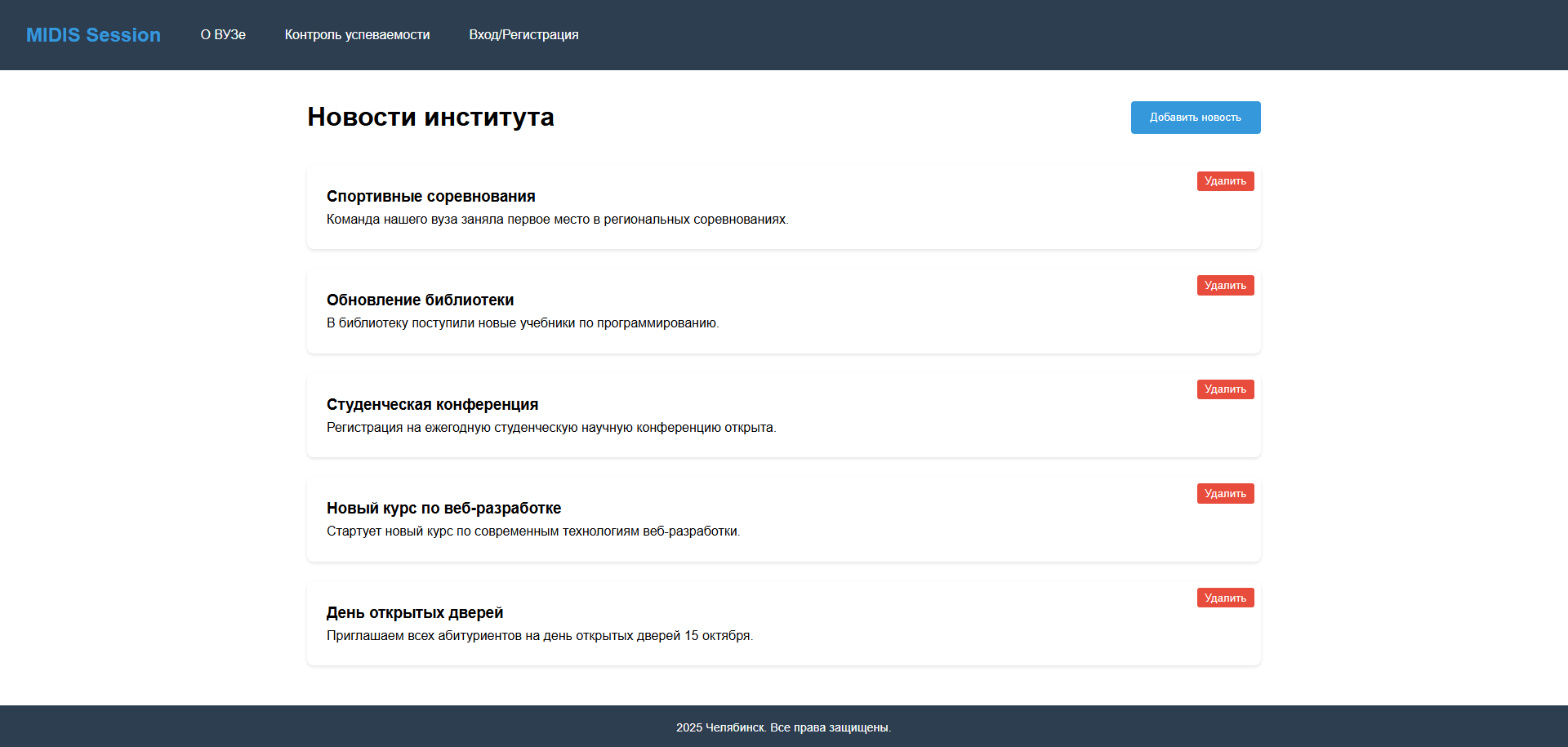


Рис. 12 – Дизайн приветственной страницы приложения информационной системы для контроля сессионной успеваемости студентов ВУЗа

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.8 Проектирование и разработка информационной системы для организации дипломного проектирования в ВУЗе

Сначала я спроектировал базу данных для данного сервиса (см. рис. 13).

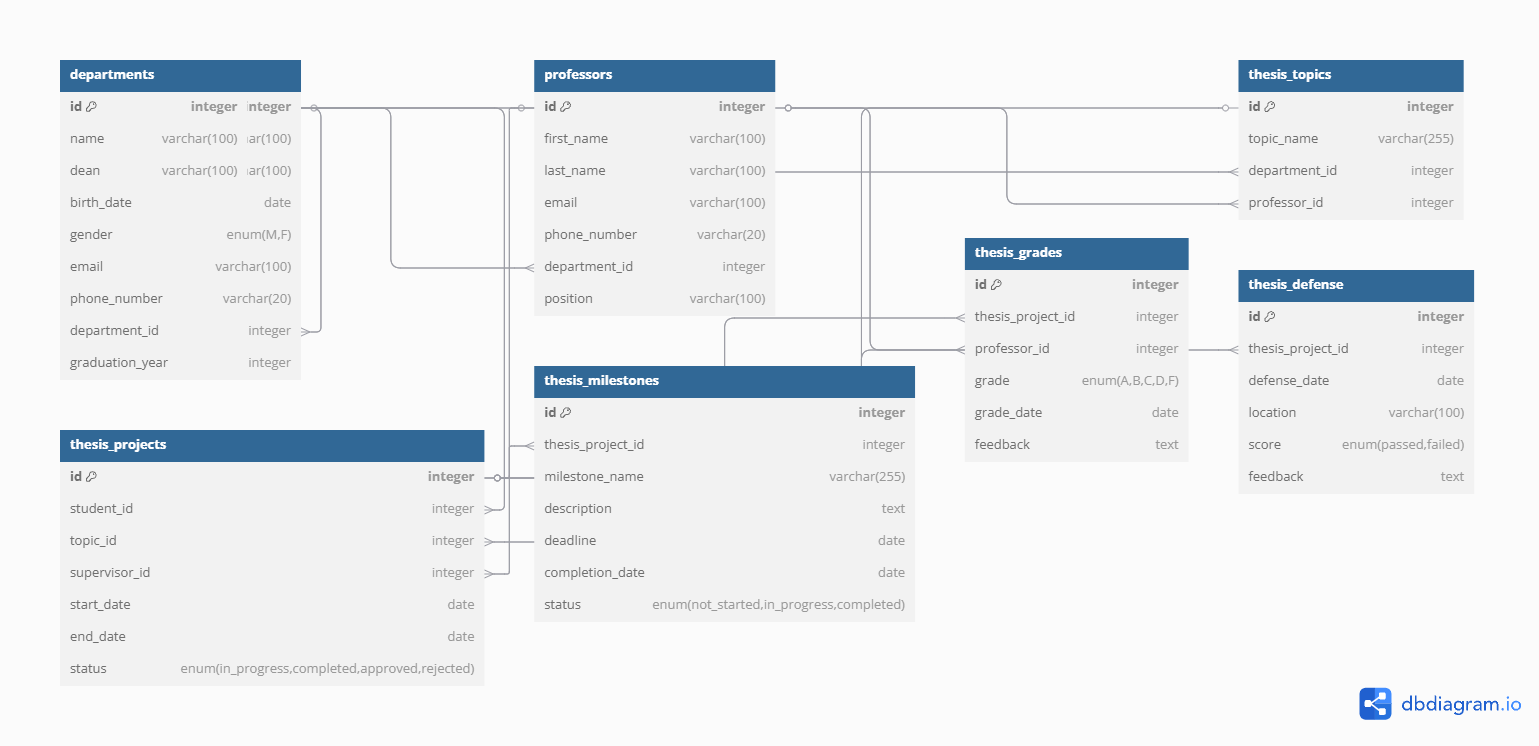


Рис. 13 - База данных для информационной системы для организации дипломного проектирования ВУЗа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

1. **Студент:**
   * Основной пользователь системы, взаимодействующий с ней для просмотра своих оценок, успеваемости и статистики.
2. **Преподаватель:**
   * Пользователь, который выставляет оценки студентам по дисциплинам и формирует отчеты по успеваемости.
3. **Администратор:**
   * Ответственный за управление данными системы: добавление, редактирование и удаление студентов, групп, дисциплин, преподавателей и сессий.
4. **Деканат:**
   * Пользователь, который анализирует и просматривает отчеты по успеваемости студентов, оценивает статистику и следит за результатами учебного процесса.
5. **Система уведомлений:**
   * Внешняя система, интегрированная с текущей системой для отправки уведомлений студентам и преподавателям (например, о новых оценках, начале сессии и важных изменениях).

**Прецеденты:**

1. **Управление профилем студента:**
   * Студент может просматривать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные студентов.
2. **Управление группами:**
   * Администратор может добавлять, редактировать или удалять группы.
3. **Управление дисциплинами:**
   * Администратор может добавлять, редактировать или удалять дисциплины.
4. **Управление преподавателями:**
   * Администратор может добавлять, редактировать или удалять данные преподавателей.
5. **Выставление оценок:**
   * Преподаватель может выставлять оценки студентам по дисциплинам.
   * Студент может просматривать свои оценки.
6. **Управление сессиями:**
   * Администратор может создавать и управлять сессиями (например, осенняя сессия 2023).
7. **Формирование отчетов:**
   * Преподаватель может формировать отчеты по успеваемости студентов.
   * Деканат может просматривать и анализировать отчеты по успеваемости.
8. **Просмотр статистики:**
   * Деканат может просматривать статистику успеваемости студентов (например, средний балл по группам или дисциплинам).
   * Студент может просматривать свою статистику (например, средний балл за сессию).
9. **Интеграция с системой уведомлений:**
   * Система уведомлений может отправлять уведомления студентам и преподавателям (например, о новых оценках, начале сессии, изменениях в расписании).

**Связи между актерами и прецедентами:**

1. **Студент:**
   * Управление профилем студента (просмотр).
   * Просмотр оценок.
   * Просмотр статистики.
2. **Преподаватель:**
   * Выставление оценок.
   * Формирование отчетов.
3. **Администратор:**
   * Управление профилем студента (добавление, редактирование, удаление).
   * Управление группами.
   * Управление дисциплинами.
   * Управление преподавателями.
   * Управление сессиями.
4. **Деканат:**
   * Формирование отчетов (анализ).
   * Просмотр статистики.
5. **Система уведомлений:**
   * Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 14).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 14 – Дизайн приветственной страницы приложения информационной системы для организации дипломного проектирования ВУЗа

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.9 Проектирование и разработка информационной системы для студенческого профкома ВУЗа

Сначала я спроектировал базу данных для данного сервиса (см. рис. 15).

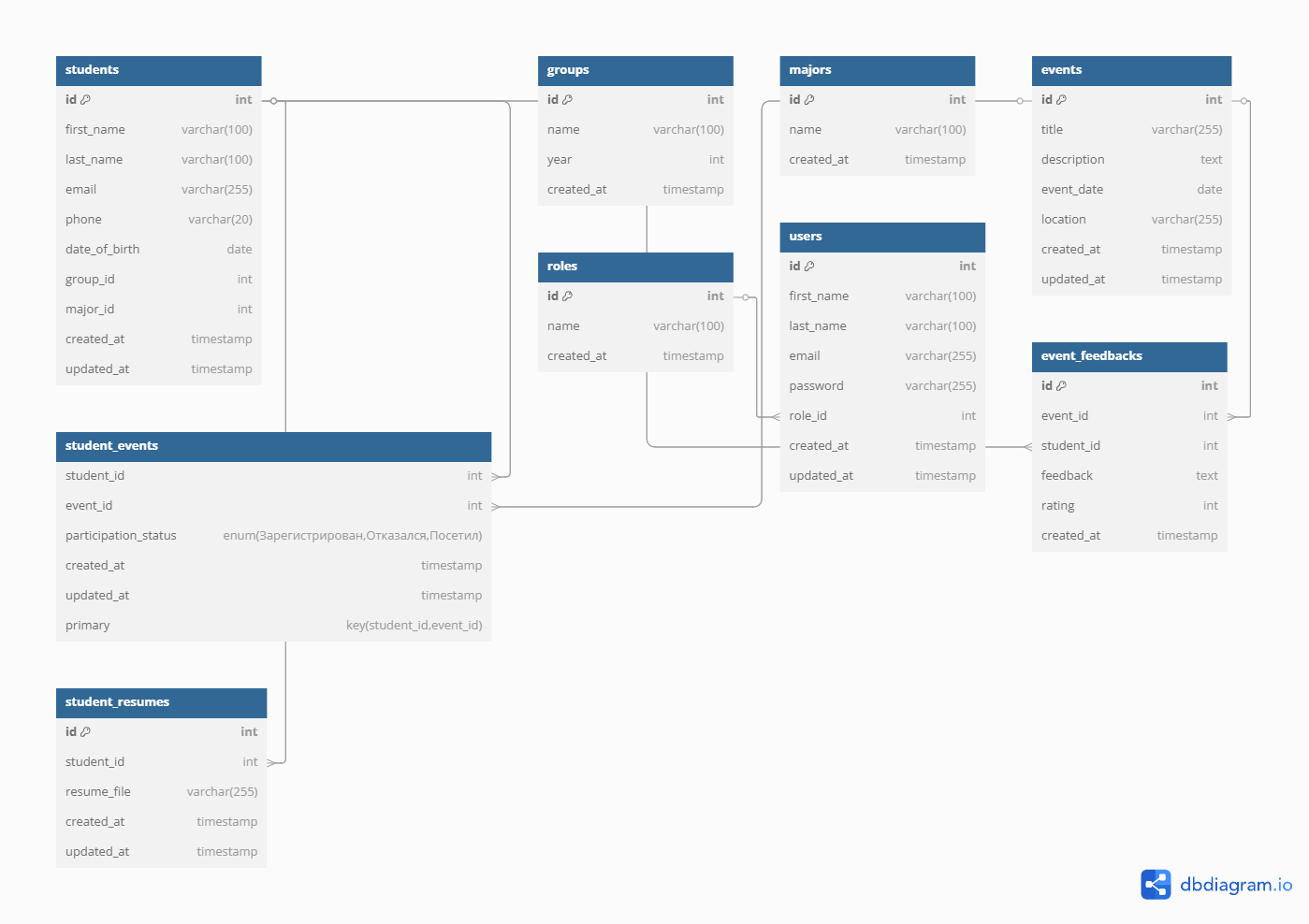


Рис. 15 - База данных для информационной системы для студенческого профкома ВУЗа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

1. Студент:
   * Основной пользователь системы, который взаимодействует с ней для участия в мероприятиях, подачи отзывов, загрузки резюме и просмотра статистики.
2. Член профкома (или администратор профкома):
   * Пользователь, который управляет мероприятиями, проверяет участие студентов и отзывы, а также анализирует статистику по различным активностям.
3. Администратор:
   * Ответственный за добавление и редактирование данных студентов, мероприятий, отзывов и других элементов системы.
4. Деканат/Руководство:
   * Пользователь, который просматривает отчеты о деятельности профкома, анализирует статистику по мероприятиям и участвует в принятии решений о поддержке активностей.
5. Система уведомлений:
   * Внешняя система, интегрированная с текущей системой для отправки уведомлений студентам и членам профкома (например, о предстоящих мероприятиях или изменениях).

Прецеденты:

1. Управление профилем студента:
   * Студент может просматривать и редактировать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные студентов.
2. Участие в мероприятиях:
   * Студент может регистрироваться и участвовать в мероприятиях.
   * Член профкома может просматривать список участников и организовывать мероприятия.
   * Администратор может добавлять, редактировать или удалять мероприятия.
3. Управление отзывами:
   * Студент может оставлять отзывы о мероприятиях.
   * Член профкома может просматривать, одобрять или отклонять отзывы.
4. Загрузка резюме:
   * Студент может загрузить свое резюме в систему.
   * Член профкома может просматривать и управлять резюме студентов.
5. Формирование отчетов:
   * Член профкома может формировать отчеты о мероприятиях, их участниках и активности.
   * Деканат/Руководство может просматривать и анализировать отчеты.
6. Просмотр статистики:
   * Деканат/Руководство может просматривать статистику по участию студентов в мероприятиях.
   * Студент может просматривать свою статистику по мероприятиям, в которых он принимал участие.
7. Интеграция с системой уведомлений:
   * Система уведомлений может отправлять уведомления студентам и членам профкома о новых мероприятиях, изменениях в расписаниях и других важных событиях.

Связи между актерами и прецедентами:

1. Студент:
   * Управление профилем студента.
   * Участие в мероприятиях.
   * Управление отзывами.
   * Загрузка резюме.
   * Просмотр статистики.
2. Член профкома:
   * Участие в мероприятиях (организация, просмотр участников).
   * Управление отзывами (просмотр, одобрение, отклонение).
   * Формирование отчетов.
   * Просмотр статистики.
3. Администратор:
   * Управление профилем студента.
   * Управление мероприятиями.
   * Управление отзывами.
   * Управление резюме.
4. Деканат/Руководство:
   * Формирование отчетов (анализ).
   * Просмотр статистики.
5. Система уведомлений:
   * Интеграция с внешней системой.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 16).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 16 – Дизайн приветственной страницы приложения информационной системы для студенческого профкома ВУЗа

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.10 Проектирование и разработка информационной системы для работника склада (вариант: склад оптовой торговли организации)

Сначала я спроектировал базу данных для данного сервиса (см. рис. 17).

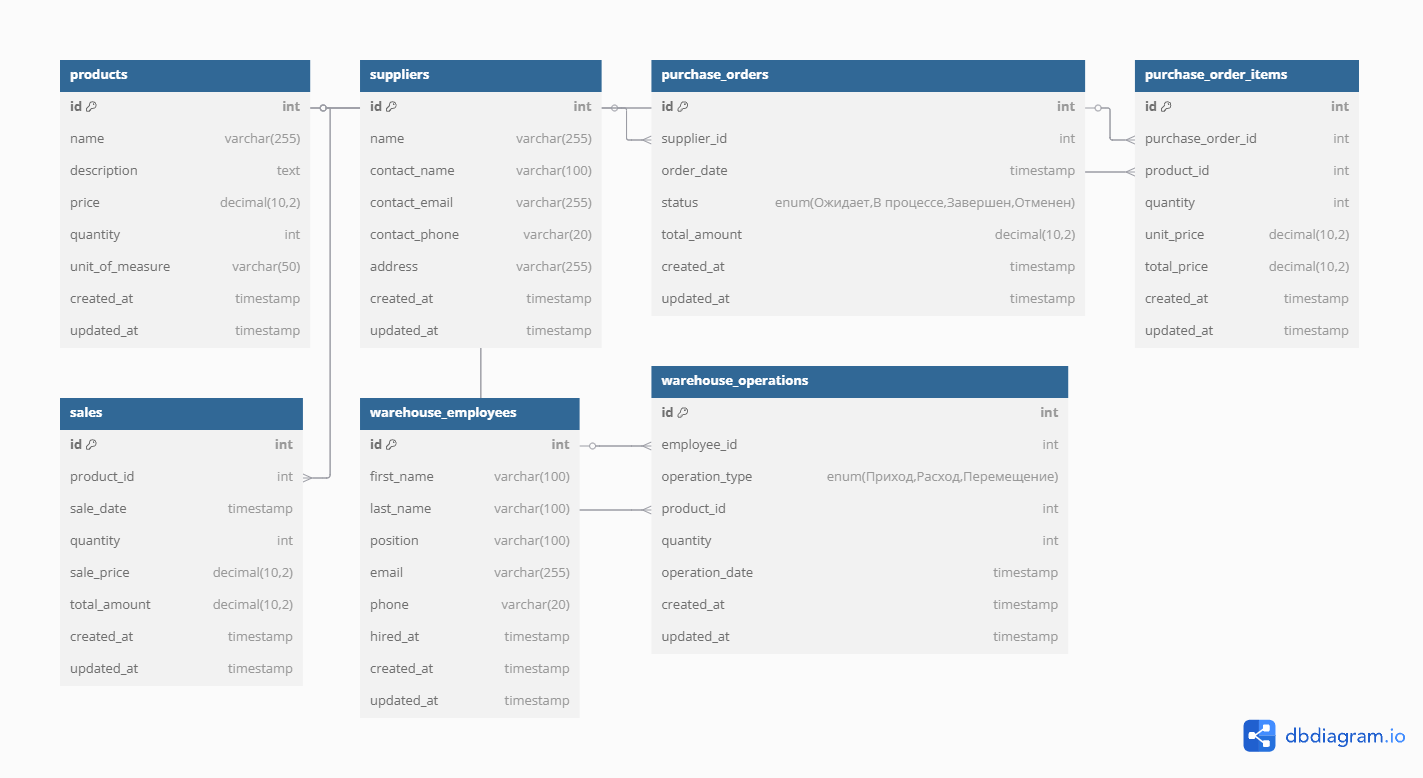


Рис. 17 - База данных для информационной системы для работника склада (вариант: склад оптовой торговли организации)

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

1. **Работник склада:**
   * Основной пользователь системы, который взаимодействует с ней для учета, приемки и отгрузки товаров, а также для выполнения складских операций.
2. **Менеджер склада:**
   * Пользователь, который управляет процессами на складе, контролирует запасы, отвечает за назначение операций и распределение задач между работниками склада.
3. **Администратор:**
   * Ответственный за настройку и управление данными системы (добавление, редактирование или удаление товаров, сотрудников склада, поставок, продаж и операций).
4. **Поставщик:**
   * Внешний партнер, который поставляет товары в организацию. Может быть интегрирован с системой для автоматического создания заказов и учета поставок.
5. **Система уведомлений:**
   * Внешняя система, которая интегрируется с текущей системой для отправки уведомлений сотрудникам склада и менеджерам о важных событиях, таких как поступление товаров, выполнение заказов и инвентаризация.

**Прецеденты:**

1. **Управление профилем работника склада:**
   * Работник склада может просматривать свои личные данные.
   * Администратор может добавлять, редактировать или удалять данные работников склада.
2. **Управление товарами:**
   * Администратор может добавлять, редактировать или удалять товары.
   * Работник склада может просматривать информацию о товарах на складе.
3. **Управление заказами:**
   * Менеджер склада может создавать, редактировать или отменять заказы.
   * Работник склада может просматривать и выполнять заказы, связанные с отгрузкой товаров.
4. **Учет поступлений и отгрузок:**
   * Работник склада может регистрировать поступление товаров на склад и их отгрузку покупателям.
   * Менеджер склада контролирует все поступления и отгрузки.
5. **Инвентаризация:**
   * Работник склада может проводить инвентаризацию товаров на складе.
   * Администратор может просматривать результаты инвентаризации и анализировать состояние складских запасов.
6. **Формирование отчетов:**
   * Менеджер склада может формировать отчеты по товарам, заказам и складу.
   * Администратор может просматривать и анализировать отчеты по товарам, поступлениям и отгрузкам.
7. **Просмотр статистики:**
   * Менеджер склада может просматривать статистику по складу, таким как объемы поступлений и отгрузок товаров.
   * Работник склада может просматривать свою рабочую статистику (например, количество выполненных операций).
8. **Интеграция с системой уведомлений:**
   * Система уведомлений может отправлять уведомления работникам склада и менеджерам о поступлении товаров, изменениях в заказах, завершении инвентаризации и других событиях.

**Связи между актерами и прецедентами:**

1. **Работник склада:**
   * Управление профилем работника склада.
   * Управление товарами (просмотр).
   * Учет поступлений и отгрузок.
   * Инвентаризация.
   * Просмотр статистики.
2. **Менеджер склада:**
   * Управление заказами.
   * Учет поступлений и отгрузок.
   * Формирование отчетов.
   * Просмотр статистики.
3. **Администратор:**
   * Управление профилем работника склада.
   * Управление товарами.
   * Формирование отчетов (анализ).
   * Управление складскими операциями.
   * Интеграция с внешней системой уведомлений.
4. **Поставщик:**
   * Интеграция с системой для поступления товаров и автоматического создания заказов.
5. **Система уведомлений:**
   * Интеграция с внешней системой для отправки уведомлений работникам склада и менеджерам.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 18).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 18 – Дизайн приветственной страницы приложения для работника склада (вариант: склад оптовой торговли организации)

После, я создал репозиторий на GitHub и настроил рабочее окружение. Затем еще раз просмотрел структуру базы данных и обдумал серверную часть. После этого создал интерфейс, с помощью выше упомянутых технологий. Код проекта регулярно коммитился в репозиторий с осмысленными комментариями. По завершении разработки провел тестирование, исправил найденные ошибки и оптимизировал код.

## 2.11 Проектирование и разработка информационной системы для библиотеки ВУЗа

Сначала я спроектировал базу данных для данного сервиса (см. рис. 19).

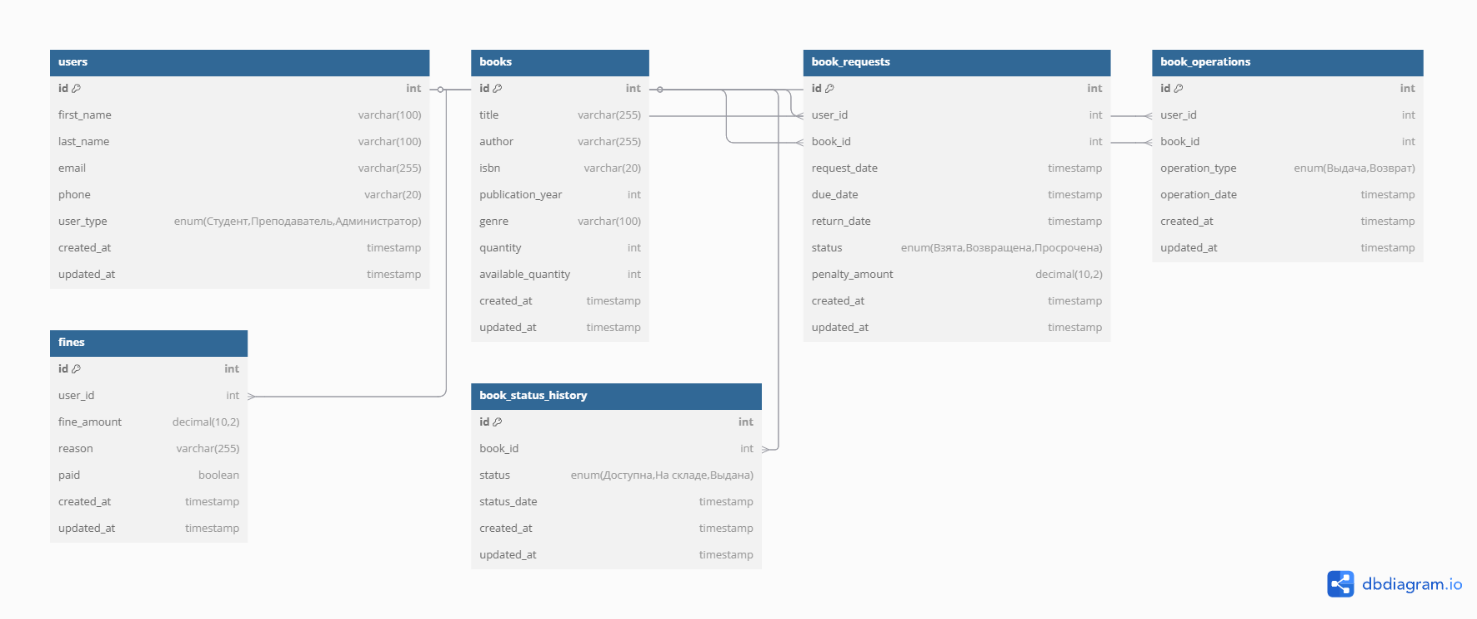


Рис. 19 - База данных для информационной системы для библиотеки ВУЗа

Далее, я приступил к заданию, связанному с формированием отношений между актерами и прецендентами для того, чтобы обозначить роль и возможности каждого пользователя.

Описание:

**Актеры:**

**Студент:**

Основной пользователь системы, который взаимодействует с ней для просмотра доступных книг, получения книг на руки, возврата книг и просмотра своих штрафов за просрочку.

**Преподаватель:**

Пользователь системы, который может заимствовать книги для учебных целей, просматривать доступные ресурсы, а также отслеживать свои штрафы за просрочку.

**Администратор:**

Ответственный за управление данными системы (добавление, редактирование или удаление книг, пользователей, управление штрафами и операциями с книгами).

**Библиотекарь:**

Пользователь, который управляет выдачей и возвратом книг, а также работает с запросами студентов и преподавателей. Также может формировать отчеты по использованию книг.

**Система уведомлений:**

Внешняя система, которая может интегрироваться с текущей системой для отправки уведомлений пользователям о просрочках, новых поступлениях книг и других важных событиях.

**Прецеденты:**

**Управление профилем пользователя:**

Студент и преподаватель могут просматривать свои личные данные.

Администратор может добавлять, редактировать или удалять данные пользователей (студентов и преподавателей).

**Управление книгами:**

Администратор и библиотекарь могут добавлять, редактировать или удалять книги в базе данных.

Студент и преподаватель могут просматривать доступные книги и их состояние (например, доступность для выдачи).

**Запросы на книги (выдача и возврат):**

Студент и преподаватель могут заимствовать книги на определенный срок.

Библиотекарь может регистрировать выдачу и возврат книг.

**Управление штрафами:**

Библиотекарь может начислять штрафы за просроченные книги.

Студент и преподаватель могут просматривать свои штрафы.

Администратор может управлять суммами штрафов и их статусом (оплачен или нет).

**Формирование отчетов по книгам:**

Библиотекарь может формировать отчеты о выдаче и возврате книг, а также об использовании книжного фонда.

Администратор и деканат могут просматривать статистику по книгам и их популярности.

**Просмотр статистики:**

Библиотекарь и администратор могут просматривать статистику по операциям с книгами (например, сколько раз книга была выдана).

Студент и преподаватель могут просматривать статистику по своей деятельности (например, количество взятых книг, сумма штрафов).

**Интеграция с системой уведомлений:**

Система уведомлений может отправлять уведомления студентам и преподавателям о новых поступлениях книг, предстоящих сроках возврата и начисленных штрафах.

**Связи между актерами и прецедентами:**

**Студент:**

Управление профилем пользователя (просмотр).

Запросы на книги (выдача и возврат).

Просмотр штрафов.

Просмотр статистики.

**Преподаватель:**

Управление профилем пользователя (просмотр).

Запросы на книги (выдача и возврат).

Просмотр штрафов.

**Администратор:**

Управление профилем пользователя (добавление, редактирование, удаление).

Управление книгами (добавление, редактирование, удаление).

Управление штрафами.

Просмотр отчетов по книгам и статистики.

**Библиотекарь:**

Запросы на книги (выдача и возврат).

Управление штрафами (начисление).

Формирование отчетов.

Просмотр статистики по книгам.

**Система уведомлений:**

Интеграция с внешней системой для отправки уведомлений пользователям.

После описания диаграммы я разработал дизайн приложения. Представлю приветственную страницу программы, в качестве примера дизайна (см. рис. 20).

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 20 – Дизайн приветственной страницы приложения информационной системы для библиотеки ВУЗа

# Выводы по Главе 1

Резюмирую выполнение Главы 1, можно сказать, что я закрепил основные этапы разработки информационных систем. Были рассмотрены такие важные моменты, как сбор требований пользователей, проектирование базы данных и интерфейсов, а также создание архитектуры системы. Я изучил различные типы ИС и их применение в реальных организациях.

# Глава 3. Создание веб-приложения для изучения физики

## 3.1 Техническое задание, выбор средств разработки

Для выполнения второго задания нам было необходимо разработать образовательное веб-приложение по заказу кафедры. Также, в это задание входит разработка прототипа и дизайна приложения. Важным условием разработки было сохранение стилевой политики нашего учебного заведения МИДиС, соответственно, также было дано задание сделать анализ цветов главного сайта нашего института. Сначала, нужно было разделиться на группы по два человека, и выбрать тему. На выбор было дано несколько тем, нам приглянулась тема про Физику.

После того, как мы определились с темой нашего приложения, мы стали обдумывать наш стек. Мы решили остановиться на Laravel, как на основном фреймворке для разработки приложения. Это удобный инструмент, c относительно низким порогом входа, где уже есть огромное количество пред настроенных функций. С его помощью можно легко настроить авторизацию и базы данных, что важно для нашего будущего приложения. К тому же важно закрепить навыки в разработке приложений на Laravel, так как в скором времени нам предстоит сдать демоэкзамен, выполняя задание на данной платформе. В качестве сборки веб-сервера мы выбрали XAMPP, так как у нас уже есть опыт работы и настройки веб-сервера через этот сборщик. Для разработки клиентской части мы будем использовать стандартный Laravel шаблонизатор Blade. Он всего лишь дополнят HTML дополнительными, нужными нам функциями, поэтому он подходит нам.

## 3.2 Разработка прототипа приложения

Первым делом мы решили разработать прототип приветственной страницы нашего будущего приложения. Мы использовали стандартный HTML и CSS, для того, чтобы показать, какие элементы будут встречаться на нашей странице с приветствием. Мы посчитали, что таких средств будет достаточно для прототипа. Так же, мы сделали прототип адаптивным (см. рис. 21).

Изображение выглядит как снимок экрана, текст, Прямоугольник, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 21 – Прототип образовательного приложения

## 3.3 Анализ цветовой политики МИДиС, разработка дизайна приветственной страницы приложения.

После создания прототипа мы приступили к разработке дизайна. Так как, по условию задания, нам нужно придерживаться стилевой повестки МИДиС, нам необходимо было провести анализ цветов на их сайте.

**Анализ цветовой схемы сайта МИДиС:**

Перед началом анализа я просмотрел веб-страницы сайта МИДиС, и выделили цветовую палитру приложения (см. рис. 22)

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.**

Рис. 22 – Цветовая палитра сайта МИДиС

**1. Accent (акцентный цвет)**

* **Назначение:** Логотип, и выделения, для акцентирования на важных элементах страницы
* **Вывод:**
  + Цвет достаточно контрастный, чтобы привлекать внимание, но не вызывает визуального дисбаланса.
  + Он хорошо сочетается с main и card, что придаёт сайту гармоничное восприятие

**2. Text (цвет текста)**

* **Особенности:**
  + Основной цвет текста.
  + Main вариация текста используется на черном фоне активных кнопок и на **list-button**.
* **Вывод:**
  + Для черного фона кнопок (активное состояние) контраст с белым текстом соответствует нормам.

**3. Main (основной фон)**

* **Назначение:** Фон страницы.
* **Вывод:**
  + Он нейтрален и не конфликтует с **text** и **card**.

**4. Card (арточки, секция hero, подвал)**

* **Назначение:** Выделение блоков информации.
* **Вывод:**
  + Присутствует контраст между **card** и **main**, чтобы карточки не сливались с фоном.
  + Цвет сочетается с общей палитрой footer и не выглядет чужеродным.

**5. List-button (фон кнопок в виджетах)**

* **Особенности:**
  + Используется для кнопок с белым текстом.
* **Вывод:**
  + Наш **list-button** — темный, контраст с белым текстом соответствует нормам доступности (WCAG AA/AAA).

В процессе разработки дизайна приветственной страницы нашего приложения мы стремились учесть основные принципы, такие как минимализм, функциональность и академическая строгость. Это обеспечило гармоничное сочетание с корпоративным стилем учебного заведения, что, в свою очередь, позволяет интегрировать страницу в общий дизайн сайта без лишних усилий.

Одним из важных шагов было использование сформированной палитры цветов. Мы постарались органично распределить цвета, чтобы они не перегружали визуальное восприятие, при этом оставались яркими и заметными в нужных элементах интерфейса, таких как кнопки, заголовки и ссылки. Такой подход позволил нам не только сохранить академическую строгость, но и сделать страницу привлекательной и удобной для пользователей.

Особое внимание было уделено тому, чтобы элементы интерфейса не только выполняли свои функции, но и смотрелись эстетично. Мы уверены, что этот дизайн будет визуально привлекательным, но и удобным для пользователей, создавая при этом ощущение единства с основным сайтом учебного заведения (см. рис. 22)Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 22 – Дизайн приветственной страницы

## 3.4 Разработка базы данных

Для начала нам нужно было выбрать диалект SQL для разработки базы данных. Мы решили остановиться на mySQL, так как он очень удобно встроен в наш фреймворк для разработки. Также, у меня имеется опыт работы с этим языком, и мы располагаем огромным количеством справочного материала в сети Интернет в других источниках.

Для грамотного построения базы данных необходимо определиться с количеством сущностей, и их связями.

В ходе размышлений мы пришли к выводу, что в нашей базе данных будет 5 сущностей:

**1. Пользователи (users)**

**Поля**:

id — уникальный идентификатор (первичный ключ).

username — уникальное имя пользователя.

email — уникальный email.

password\_hash — хеш пароля.

created\_at — дата регистрации.

**Связи**:

Связана с user\_answers через user\_id (один пользователь → много ответов).

**2. Разделы (sections)**

**Поля**:

id — уникальный идентификатор.

name — название раздела (например, "Математика", "Физика").

description — описание раздела.

**Связи**:

Связана с topics через section\_id (один раздел → много тем).

**3. Темы (topics)**

**Поля**:

id — уникальный идентификатор.

section\_id — ссылка на раздел (внешний ключ).

name — название темы (например, "Алгебра", "Механика").

order\_position — порядок темы внутри раздела.

**Связи**:

Принадлежит к одному разделу (sections).

Связана с problems через topic\_id (одна тема → много задач).

**4. Задачи (problems)**

**Поля**:

id — уникальный идентификатор.

topic\_id — ссылка на тему (внешний ключ).

question — текст задачи.

correct\_answer — правильный ответ.

difficulty — уровень сложности (easy, medium, hard).

explanation — объяснение решения.

**Связи**:

Принадлежит к одной теме (topics).

Связана с user\_answers через problem\_id (одна задача → много ответов).

**5. Ответы пользователей (user\_answers)**

**Поля**:

id — уникальный идентификатор.

user\_id — ссылка на пользователя (внешний ключ).

problem\_id — ссылка на задачу (внешний ключ).

user\_answer — ответ пользователя.

is\_correct — флаг правильности ответа.

created\_at — время ответа.

**Связи**:

Связана с users и problems (многие ответы → один пользователь/задача).

Итоговая схема базы данных: (см. рис. 23)

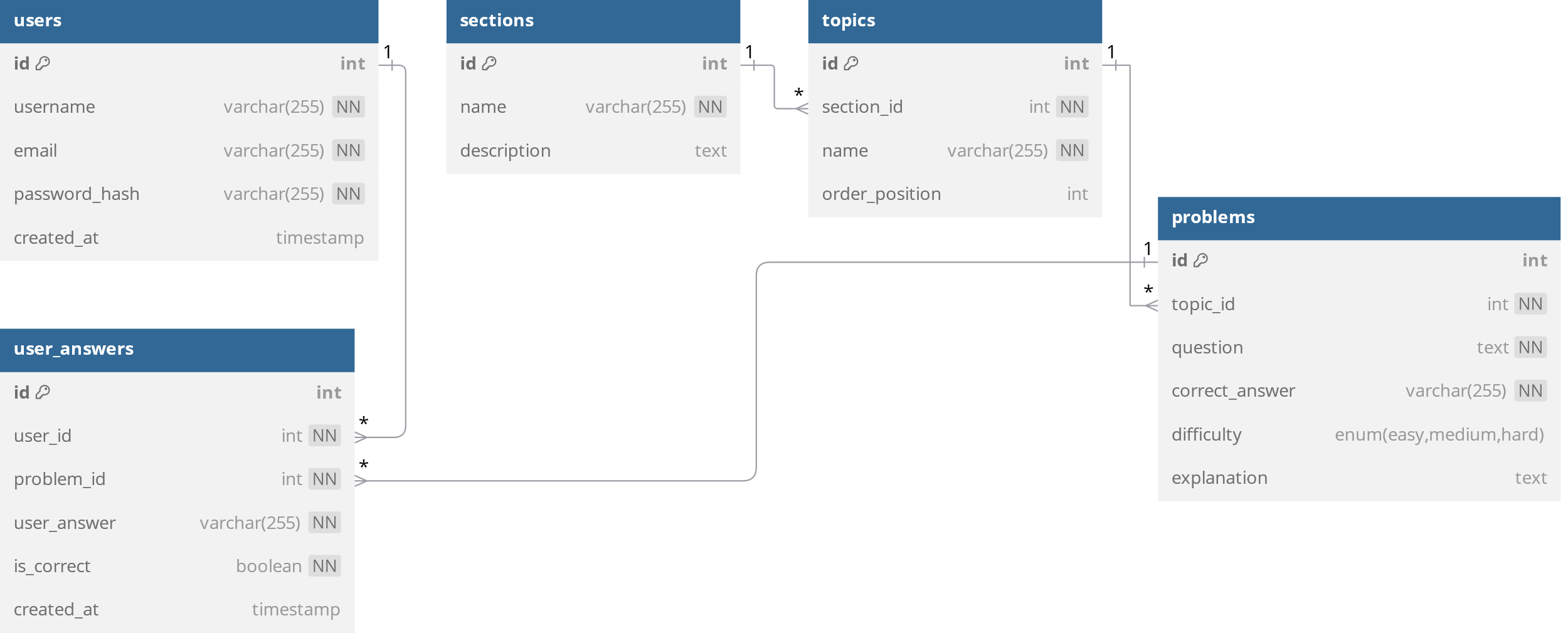


Рис. 23 – Итоговая схема базы данных

# Приложение

Таблица 1 – Соотношение тем и выполненных работ для Главы 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № Варианта | Тема | Имя файла макета | Ссылка на макет | Ссылка на сайт | Ссылка на репозиторий |
| 1 | Проектирование и разработка информационной системы для контроля выполнения нагрузки преподавателей ВУЗа. | 1.fig | <https://www.figma.com/design/m0uz4EiFOUzfuZdORYNyLu/1?node-id=1669-162202&p=f&t=onnEsDiME9Rx9gmz-0> | [**https://immoee.github.io/-1/**](https://immoee.github.io/-1/) | <https://github.com/immoEe/-1> |
| 2 | Проектирование и разработка информационной системы для работника склада (вариант: склад торговой организации, занимающейся сбытом как продукции собственного производства, так и продукции внешних поставщиков;) | 2 вариант.fig | <https://www.figma.com/design/IU2I7nOB1SwzwYdippluIv/2-%D0%B2%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82?node-id=0-1&p=f&t=FqECzikeGN2h4WYt-0> | [[**https://immoee.github.io/-2/**](https://immoee.github.io/-2/)f](https://www.figma.com/design/IU2I7nOB1SwzwYdippluIv/2-%D0%B2%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82?node-id=0-1&p=f) | <https://github.com/immoEe/-2> |
| 3 | Проектирование и разработка информационной системы детского сада. | 3 вариант.fig | <https://www.figma.com/design/hXl4sDPQmKueRHtFhVm7Bv/3-%D0%B2%D0%B0%D1%80%D0%B8%D0%B0%D0%BD%D1%82> | [**https://immoee.github.io/-3/**](https://immoee.github.io/-3/) | <https://github.com/immoEe/-3> |
| 4 | Проектирование и разработка информационной системы для управления работой тренажерного зала свободного доступа. | 4.fig | <https://www.figma.com/design/kKbhlSMJaAaANaDicAjBOg/4?node-id=0-1&p=f&t=onnEsDiME9Rx9gmz-0> | [**https://immoee.github.io/-4/**](https://immoee.github.io/-4/) | <https://github.com/immoEe/-4> |
| 5 | Проектирование и разработка информационной системы для домашней библиотеки. | 5.fig | <https://www.figma.com/design/276Ln3tGBNIGlBSwU49Hs0/Untitled?node-id=0-1&p=f&t=d6eD4xmr7jd2WcF4-0> | [**https://immoee.github.io/-5/**](https://immoee.github.io/-5/) | <https://github.com/immoEe/-5> |
| 6 | Проектирование и разработка информационной системы для контроля сессионной успеваемости студентов ВУЗа. | 6.fig | <https://www.figma.com/design/SSXGK0fknn87BDONtTgtcl/6?node-id=0-1&p=f&t=d6eD4xmr7jd2WcF4-0> | [**https://immoee.github.io/-6/**](https://immoee.github.io/-6/) | <https://github.com/immoEe/-6> |
| 7 | Проектирование и разработка информационной системы для организации дипломного проектирования в ВУЗе. | 7.fig | <https://www.figma.com/design/CYarsUans8YYs8pCBBWyCo/7?node-id=0-1&p=f&t=VxxgunZmB7yUIyNz-0> | [**https://immoee.github.io/-7/**](https://immoee.github.io/-7/) | <https://github.com/immoEe/-7> |
| 8 | Проектирование и разработка информационной системы для студенческого профкома ВУЗа. | 8.fig | <https://www.figma.com/design/4VencSjYHrzhpbF6J3aQcn/Untitled?node-id=0-1&p=f&t=VPlTADCvtl1Gxvrs-0> | [**https://immoee.github.io/-8/**](https://immoee.github.io/-8/) | <https://github.com/immoEe/-8> |
| 9 | Проектирование и разработка информационной системы для работника склада (вариант: склад оптовой торговой организации;) | 9.fig | <https://www.figma.com/design/Np69Rijl6jBBu61aj6L2u1/7?node-id=0-1&p=f&t=d6eD4xmr7jd2WcF4-0> | [**https://immoee.github.io/-9/**](https://immoee.github.io/-9/) | <https://github.com/immoEe/-9> |
| 10 | Проектирование и разработка информационной системы для библиотеки ВУЗа. | 10.fig | <https://www.figma.com/design/vfNbfdmFJPnrJnYUUzFvsn/Untitled?node-id=0-1&p=f&t=h4MVZUYX9bZK0KTs-0> | [**https://immoee.github.io/-10/**](https://immoee.github.io/-10/) | <https://github.com/immoEe/-10> |