ФГБОУ ВПО Уральский государственный горный университет

Инженерно-экономический факультет

Кафедра информ@тики

Курсовой проект

По дисциплине «Технологии программирования» На тему «Разработка приложения “Онлайн-система управления учебной нагрузкой преподавателей”»

Выполнил:

Студент гр. АУБП-22-2

Бруевич М.А.

Проверила:

ст. преп. каф. Информатики

Волкова Е.А.

Екатеринбург, 2025г.

1. Постановка задачи
   1. Характеристика объекта и проблемной области

Объект автоматизации:

Процесс планирования, распределения и учета учебной нагрузки преподавателей в высшем учебном заведении.

Проблемная область:

* + В большинстве вузов учет учебной нагрузки преподавателей ведется вручную или с помощью электронных таблиц. Это приводит к следующим проблемам:
  + Высокая вероятность ошибок при ручном вводе и обработке данных.
  + Затруднённый контроль выполнения нагрузки и формирование отчетности.
  + Отсутствие единого централизованного хранилища информации.
  + Сложности в оперативном доступе к актуальным данным для преподавателей и руководства.
  + Трудоемкость процесса согласования и перераспределения нагрузки.

Существующие варианты решения:

* Использование электронных таблиц (Excel, Google Sheets) — не обеспечивает автоматизации, требует ручного контроля и не защищает данные.
* Корпоративные системы учета (например, 1С:Университет) — часто избыточны по функционалу, сложны в освоении и дорогостоящи.
* Бумажные журналы и ведомости — полностью ручной труд, высокая вероятность ошибок, отсутствие прозрачности.
  1. Постановка целей и задач автоматизации/информатизации

Основная цель автоматизации:

Создание онлайн-системы, обеспечивающей автоматизированное планирование, учет и контроль учебной нагрузки преподавателей, а также формирование отчетности для руководства.

Задачи, решаемые системой:

* + Централизованный учет учебной нагрузки преподавателей по дисциплинам, группам, видам занятий и учебным годам.
  + Автоматизация процесса назначения и перераспределения нагрузки заведующими кафедрами.
  + Предоставление преподавателям возможности оперативно просматривать свою нагрузку.
  + Формирование отчетов по выполненной нагрузке для руководства кафедры и администрации. Обеспечение разграничения прав доступа в зависимости от роли пользователя (администратор, заведующий кафедрой, преподаватель).
  + Снижение количества ошибок и трудозатрат, связанных с ручным учетом и отчетностью.
  1. Краткое техническое задание

Наименование системы: Онлайн-система управления учебной нагрузкой преподавателей

Основание для разработки: Необходимость автоматизации процессов планирования и учета учебной нагрузки в образовательной организации.

Требования к системе (по FURPS+):

F (Functionality — функциональность):

* + Система должна обеспечивать аутентификацию и авторизацию пользователей с ролями: администратор, заведующий кафедрой, преподаватель.
  + Система должна позволять заведующему кафедрой назначать нагрузку преподавателям своей кафедры.
  + Система должна хранить сведения о преподавателях, дисциплинах, учебных группах, учебной нагрузке.
  + Система должна формировать отчеты по выполненной нагрузке за выбранный учебный год и экспортировать их в формате CSV.
  + Система должна позволять преподавателю просматривать только свою нагрузку.

U (Usability — удобство использования):

* + Интерфейс должен быть реализован на русском языке.
  + Все основные действия должны быть доступны не более чем в 3 клика от главной страницы.
  + Время отклика интерфейса на действия пользователя не должно превышать 2 секунд при стандартной нагрузке.
  + Интерфейс должен быть выполнен в фирменных цветах университета: основной — #003366, дополнительный — #FFD700.

R (Reliability — надежность):

* Система должна обеспечивать сохранность данных при сбое сервера (использование транзакций при записи в БД).
* Доступность системы — не менее 95% времени в течение учебного года.

P (Performance — производительность):

* Время генерации отчета по нагрузке не должно превышать 5 секунд при объеме данных до 1000 записей.
* Система должна поддерживать одновременную работу не менее 20 пользователей.

S (Supportability — сопровождаемость):

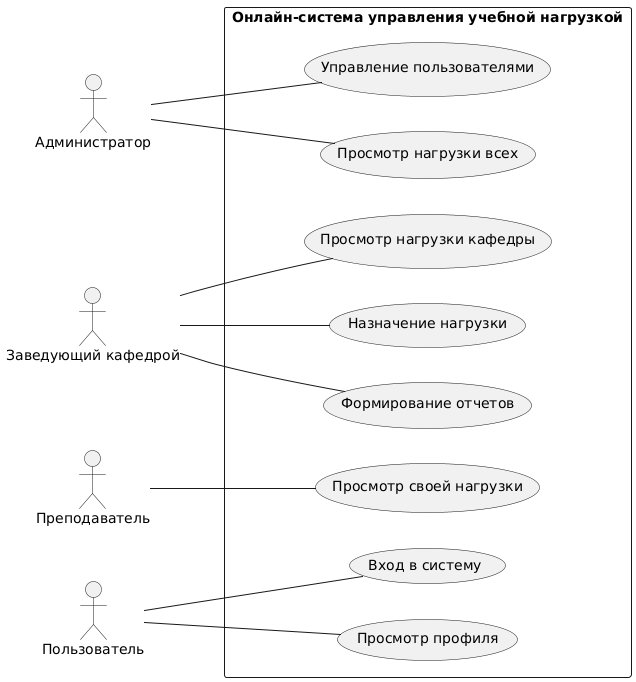
* Система должна быть реализована с использованием фреймворка Spring Boot и СУБД MySQL.
* Исходный код должен быть документирован на уровне классов и методов. В системе должна быть реализована возможность резервного копирования данных.

1. Проектирование системы
   1. Моделирование системы

UML-диаграмма вариантов использования (Use Case Diagram):

Основные варианты использования:

* Вход в систему
* Управление пользователями (создание, редактирование, удаление) — только администратор
* Просмотр и редактирование профиля
* Просмотр учебной нагрузки (все роли, но с разным уровнем доступа)
* Назначение учебной нагрузки (только заведующий кафедрой)
* Формирование и экспорт отчетов (только заведующий кафедрой)
* Просмотр статистики и дашборда

Рисунок 1. Диаграмма Use Case вариантов использования приложения “Онлайн-система управления учебной нагрузкой преподавателей”.

Контекстная (концептуальная) диаграмма:

Основные сущности:

* User — пользователь системы (админ, заведующий кафедрой, преподаватель)
* Teacher — преподаватель, связанный с User
* StudentGroup — учебная группа Discipline — дисциплина
* TeachingLoad — учебная нагрузка (связь преподавателя, дисциплины, группы, часов и типа занятия)

Связи:

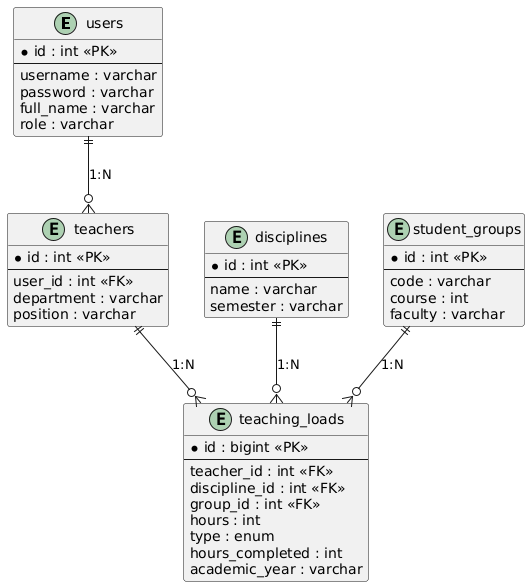
* User ↔ Teacher (один-к-одному)
* Teacher ↔ TeachingLoad (один-ко-многим)
* Discipline ↔ TeachingLoad (один-ко-многим)
* StudentGroup ↔ TeachingLoad (один-ко-многим)

User 1---1 Teacher 1---\* TeachingLoad \*---1 Discipline

|

\*---1 StudentGroup

* 1. Проектирование БД

Рисунок 2. ER-диаграмма.

Описание основных таблиц и полей:

Таблица users:

* id — уникальный идентификатор пользователя (PK)
* username — логин пользователя (уникальный)
* password — пароль пользователя
* full\_name — ФИО пользователя
* role — роль пользователя (admin, head\_of\_dept, teacher)

Таблица teachers:

* id — уникальный идентификатор преподавателя (PK)
* user\_id — внешний ключ на пользователя (FK → users.id)
* department — кафедра position — должность

Таблица disciplines:

* id — уникальный идентификатор дисциплины (PK)
* name — наименование дисциплины
* semester — семестр

Таблица student\_groups:

* id — уникальный идентификатор группы (PK)
* code — код группы (уникальный)
* course — курс
* faculty — факультет

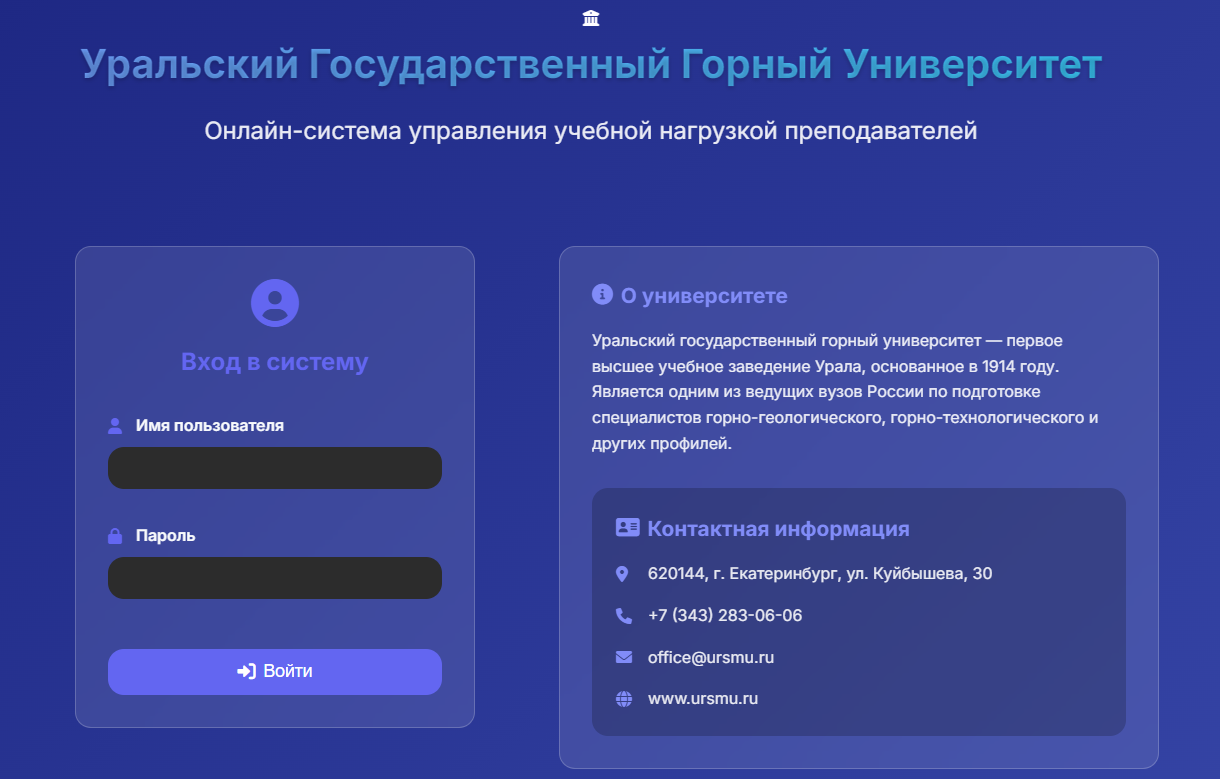
Таблица teaching\_loads:

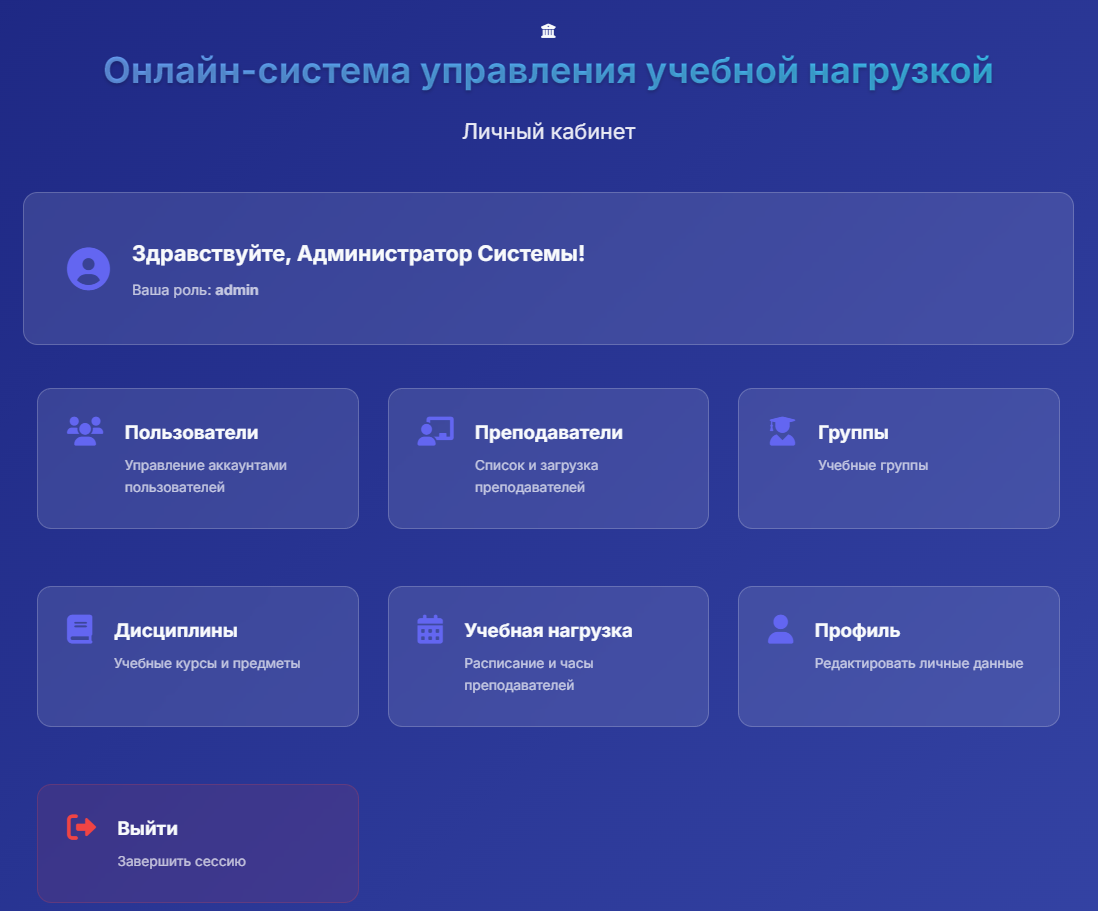
* id — уникальный идентификатор записи нагрузки (PK)
* teacher\_id — внешний ключ на преподавателя (FK → teachers.id)
* discipline\_id — внешний ключ на дисциплину (FK → disciplines.id)
* group\_id — внешний ключ на группу (FK → student\_groups.id)
* hours — количество часов по нагрузке type — тип занятия (лекция, практика, лабораторная)
* hours\_completed — количество выполненных часов
* academic\_year — учебный год

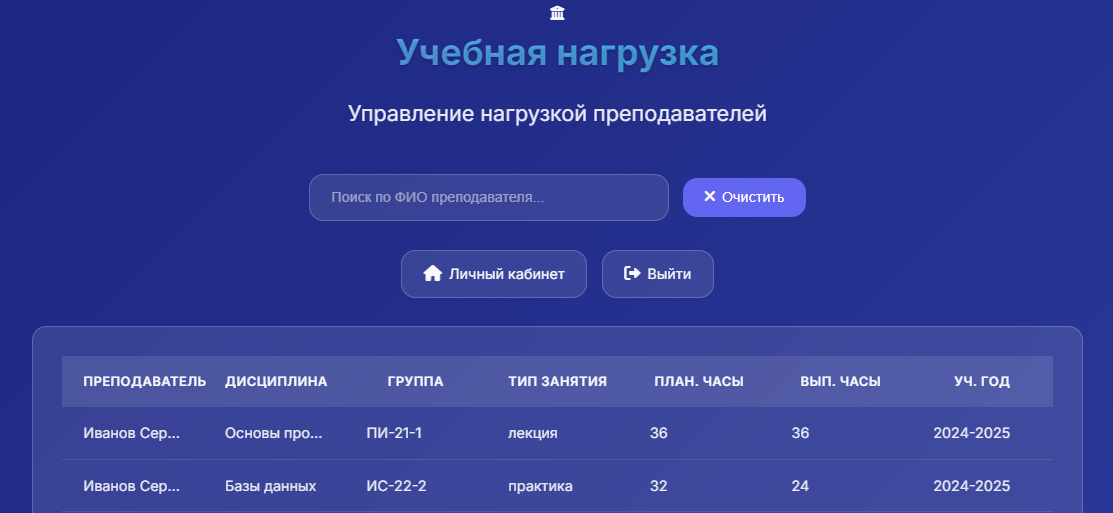
Связи между таблицами:

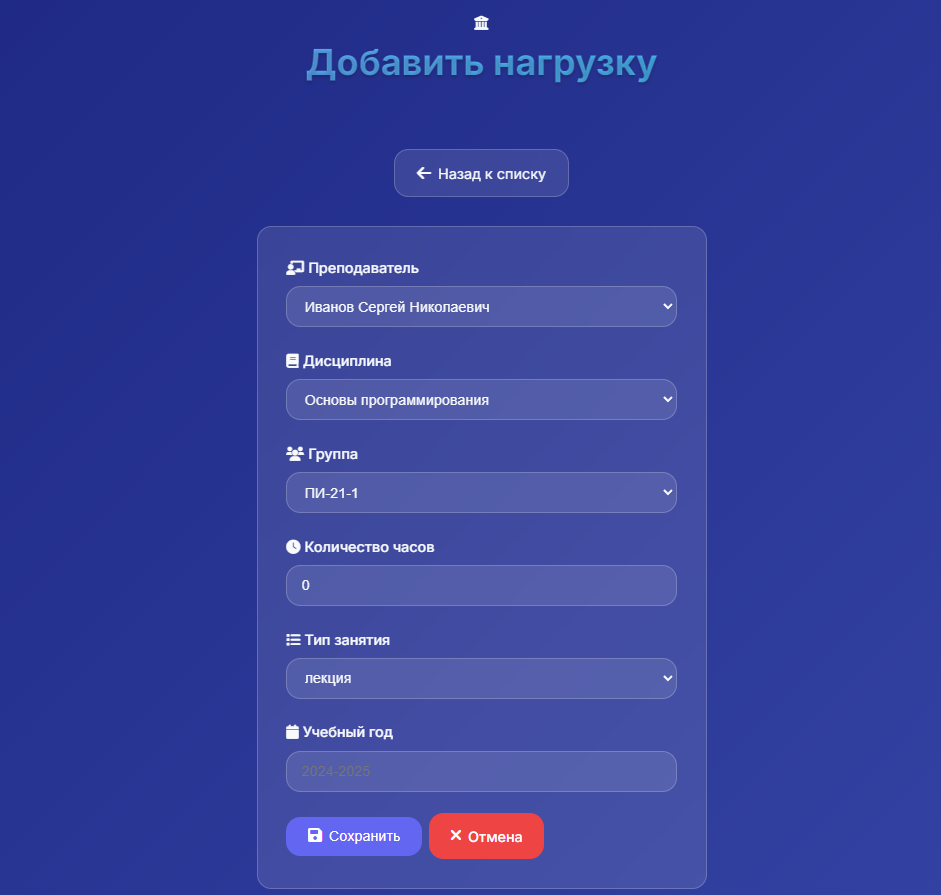
* Один пользователь связан с одним преподавателем (users 1:1 teachers)
* Один преподаватель может иметь много записей нагрузки (teachers 1:N teaching\_loads)
* Одна дисциплина может встречаться во многих нагрузках (disciplines 1:N teaching\_loads)
* Одна группа может встречаться во многих нагрузках (student\_groups 1:N teaching\_loads)
  1. Проектирование интерфейса

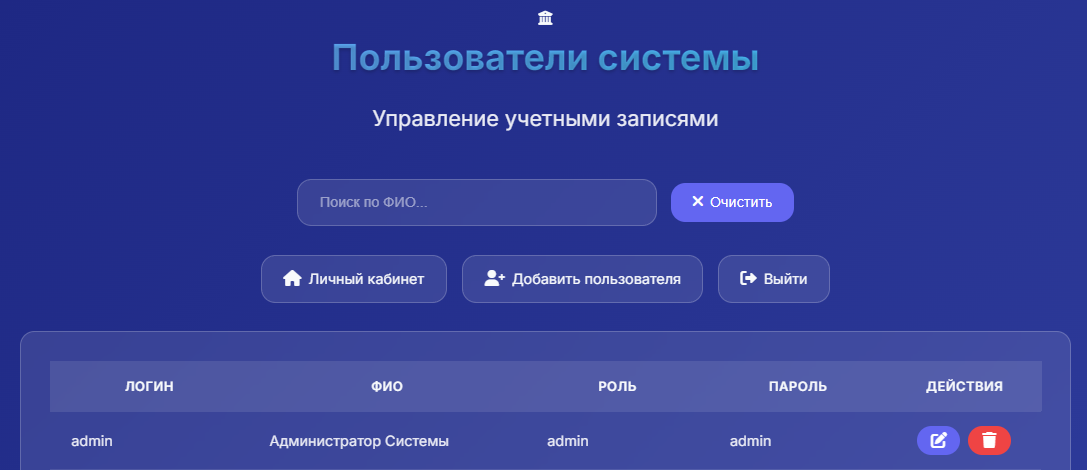
В данном разделе представлены скетчи основных экранов приложения «Онлайн-система управления учебной нагрузкой преподавателей».

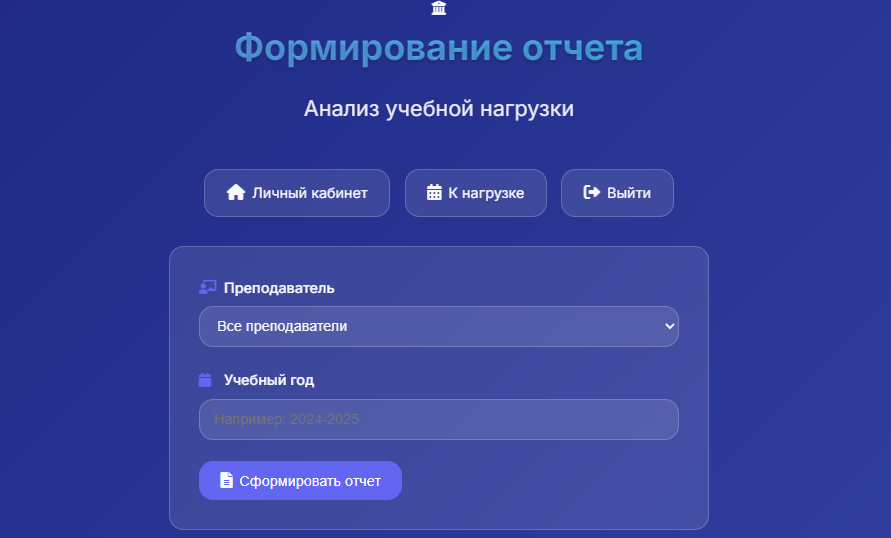
Рисунок 3. Экран входа в систему.

Рисунок 4. Главная страница пользователя (администратор).

Рисунок 5. Список учебной нагрузки (для преподавателя).

Рисунок 6. Назначение нагрузки (для заведующего кафедрой).

Рисунок 7.Управление пользователями (для администратора).

Рисунок 8. Формирование отчета (для заведующего кафедрой).

1. Разработка системы
   1. Выбор средств реализации

В качестве основного языка программирования выбран Java. Это один из самых популярных языков для разработки корпоративных и веб-приложений, обладающий широкой экосистемой, поддержкой объектно-ориентированного программирования, высокой безопасностью и масштабируемостью. Java хорошо интегрируется с современными фреймворками и СУБД, а также поддерживается большинством хостингов и облачных платформ.

Для реализации серверной части приложения используется фреймворк Spring Boot. Он позволяет быстро создавать производительные и масштабируемые веб-приложения, предоставляет встроенные средства для работы с REST, безопасностью, шаблонизацией, а также поддерживает автоматическую конфигурацию и внедрение зависимостей. Spring Boot значительно ускоряет процесс разработки и облегчает сопровождение кода.

В качестве СУБД выбрана MySQL. Это одна из самых распространённых реляционных СУБД, отличающаяся высокой производительностью, надежностью и бесплатной

лицензией. MySQL легко интегрируется с Java-приложениями через стандартные драйверы и поддерживается Spring Data JPA.

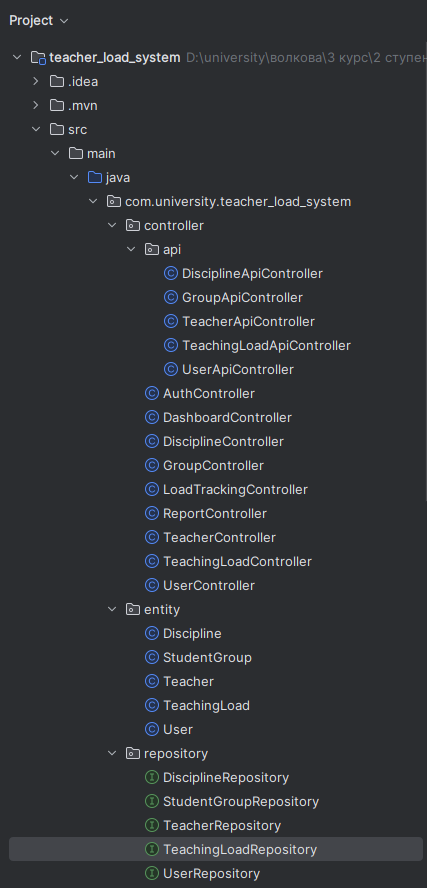
Для работы с базой данных используется Spring Data JPA, который реализует паттерн Repository и позволяет работать с данными на уровне объектов, а не SQL-запросов. Это ускоряет разработку, уменьшает количество шаблонного кода и снижает вероятность ошибок при работе с БД.

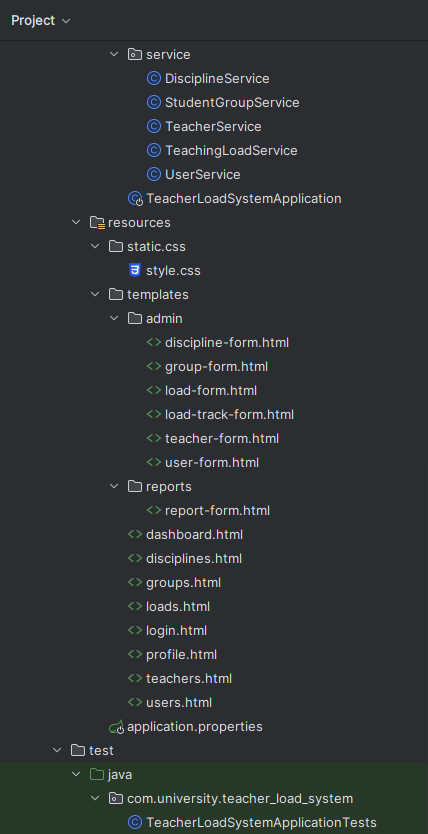
Для генерации HTML-страниц выбран шаблонизатор Thymeleaf. Он легко интегрируется с Spring Boot, поддерживает динамическую генерацию страниц, работу с формами и валидацией, а также позволяет создавать современные и адаптивные пользовательские интерфейсы.

Аргументация выбора:

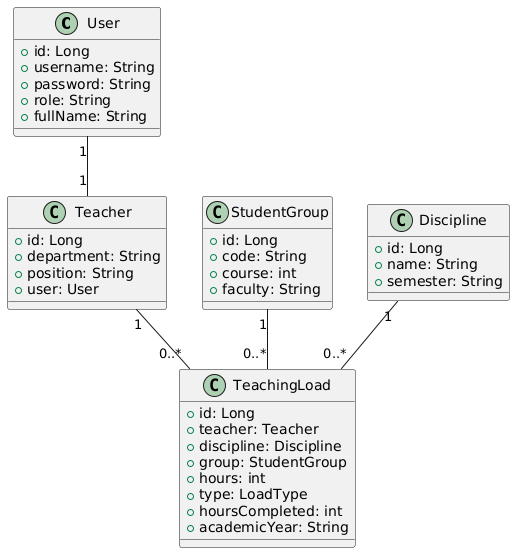
* Все выбранные технологии являются промышленными стандартами, имеют широкое сообщество и хорошую документацию.
* Использование Spring Boot и связанных с ним инструментов позволяет быстро разрабатывать, тестировать и масштабировать приложение.
* MySQL обеспечивает надежное хранение данных и легко разворачивается как на локальных, так и на облачных серверах.
* Java и Spring Boot позволяют реализовать строгую типизацию, модульность и безопасность приложения.
* Использование шаблонизатора Thymeleaf обеспечивает удобную интеграцию серверной и клиентской части, а также поддержку локализации.
  1. Структура проекта

Дерево проекта:





UML-диаграмма классов:



Краткое описание структуры решения:

* controller/ — содержит контроллеры Spring MVC, реализующие обработку HTTP-запросов и взаимодействие с пользователем.
* entity/ — содержит классы-сущности, отражающие структуру данных и таблиц БД.
* repository/ — содержит интерфейсы для доступа к данным (Spring Data JPA).
* service/ — содержит сервисы, реализующие бизнес-логику приложения.
* resources/ — содержит шаблоны представлений (Thymeleaf), файлы конфигурации и статические ресурсы.
* test/ — содержит модульные и интеграционные тесты.
  1. Реализация

**Листинги нетривиальных функций основных классов:**

Контроллер управления учебной нагрузкой (TeachingLoadController.java):

// Получение списка нагрузки для разных ролей  
@GetMapping  
public String showLoads(Model model, HttpSession session) {  
 User currentUser = (User) session.getAttribute("user");  
  
 if (currentUser == null || (!"admin".equals(currentUser.getRole())  
 && !"head\_of\_dept".equals(currentUser.getRole())  
 && !"teacher".equals(currentUser.getRole()))) {  
 return "redirect:/login";  
 }  
  
 List<TeachingLoad> loads;  
  
 if ("teacher".equals(currentUser.getRole())) {  
 // Преподаватель видит только свою нагрузку  
 Teacher teacher = teacherService.findByUser(currentUser);  
 if (teacher == null) {  
 model.addAttribute("error", "Для вашей учетной записи не создан профиль преподавателя. Обратитесь к администратору.");

model.addAttribute("loads", List.of());  
 model.addAttribute("totalHours", 0);  
 model.addAttribute("totalCompleted", 0);  
 model.addAttribute("user", currentUser);  
 model.addAttribute("userRole", currentUser.getRole());  
 return "loads";  
 }  
 loads = teachingLoadService.findByTeacherId(teacher.getId());  
 } else if ("head\_of\_dept".equals(currentUser.getRole())) {  
 // Заведующий кафедрой — нагрузка по своей кафедре  
 Teacher head = teacherService.findByUser(currentUser);  
 loads = teachingLoadService.findByTeacherDepartment(head.getDepartment());  
 } else {  
 // Админ — все данные  
 loads = teachingLoadService.findAll();  
 }  
  
 int totalHours = loads.stream().mapToInt(TeachingLoad::getHours).sum();  
 int totalCompleted = loads.stream().mapToInt(TeachingLoad::getHoursCompleted).sum();  
  
 model.addAttribute("loads", loads);  
 model.addAttribute("totalHours", totalHours);  
 model.addAttribute("totalCompleted", totalCompleted);  
 model.addAttribute("user", currentUser);  
 model.addAttribute("userRole", currentUser.getRole());  
  
 return "loads";  
}

Данный метод реализует отображение нагрузки для разных ролей пользователей: преподаватель видит только свою нагрузку, заведующий кафедрой — нагрузку по своей кафедре, администратор — всю нагрузку.

Сервис формирования отчёта (ReportController.java):

@PostMapping  
public String generateReport(@RequestParam(required = false) Long teacherId,  
 @RequestParam String academicYear,  
 Model model,  
 HttpSession session) {  
 User currentUser = (User) session.getAttribute("user");  
  
 if (currentUser == null || !"head\_of\_dept".equals(currentUser.getRole())) {  
 return "redirect:/login";  
 }  
 Teacher head = teacherService.findByUser(currentUser);  
 List<TeachingLoad> loads;  
 if (teacherId != null && teacherId > 0) {  
 loads = teachingLoadService.findByTeacherIdAndYear(teacherId, academicYear);  
 } else {  
 loads = teachingLoadService.findByTeacherDepartmentAndYear(head.getDepartment(), academicYear);  
 }  
  
 int totalHours = loads.stream().mapToInt(TeachingLoad::getHours).sum();  
 int totalCompleted = loads.stream().mapToInt(TeachingLoad::getHoursCompleted).sum();  
 model.addAttribute("loads", loads);  
 model.addAttribute("totalHours", totalHours);  
 model.addAttribute("totalCompleted", totalCompleted);  
 model.addAttribute("academicYear", academicYear);  
 model.addAttribute("teachers", teacherService.findByDepartment(head.getDepartment()));  
 model.addAttribute("selectedTeacherId", teacherId);  
 model.addAttribute("currentYear", academicYear);  
 return "reports/report-form";  
}

Метод формирует отчет по нагрузке за выбранный учебный год и преподавателя или по всей кафедре.

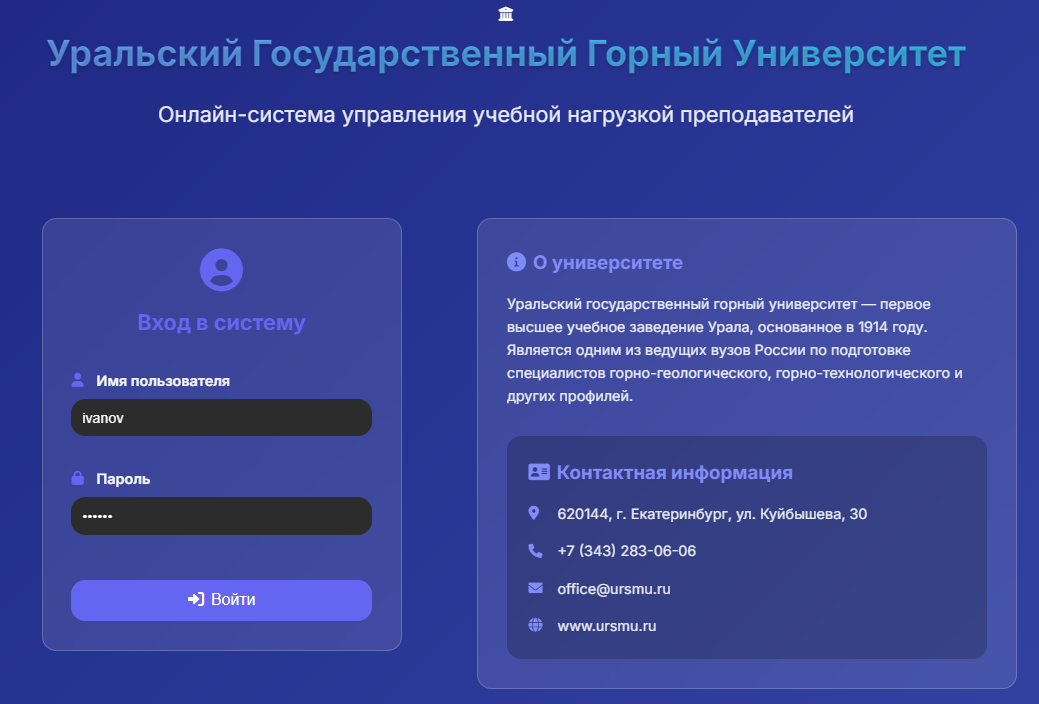
**Листинг создания БД и описание структуры данных:**

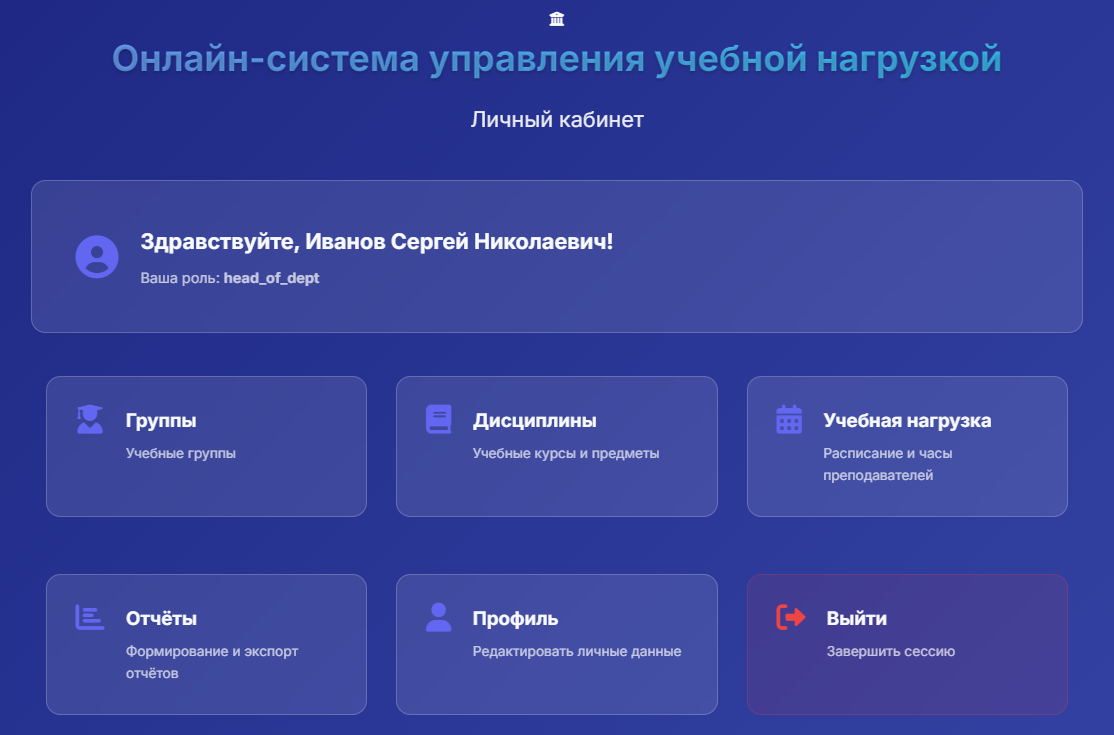
SQL для создания таблицы teaching\_loads:

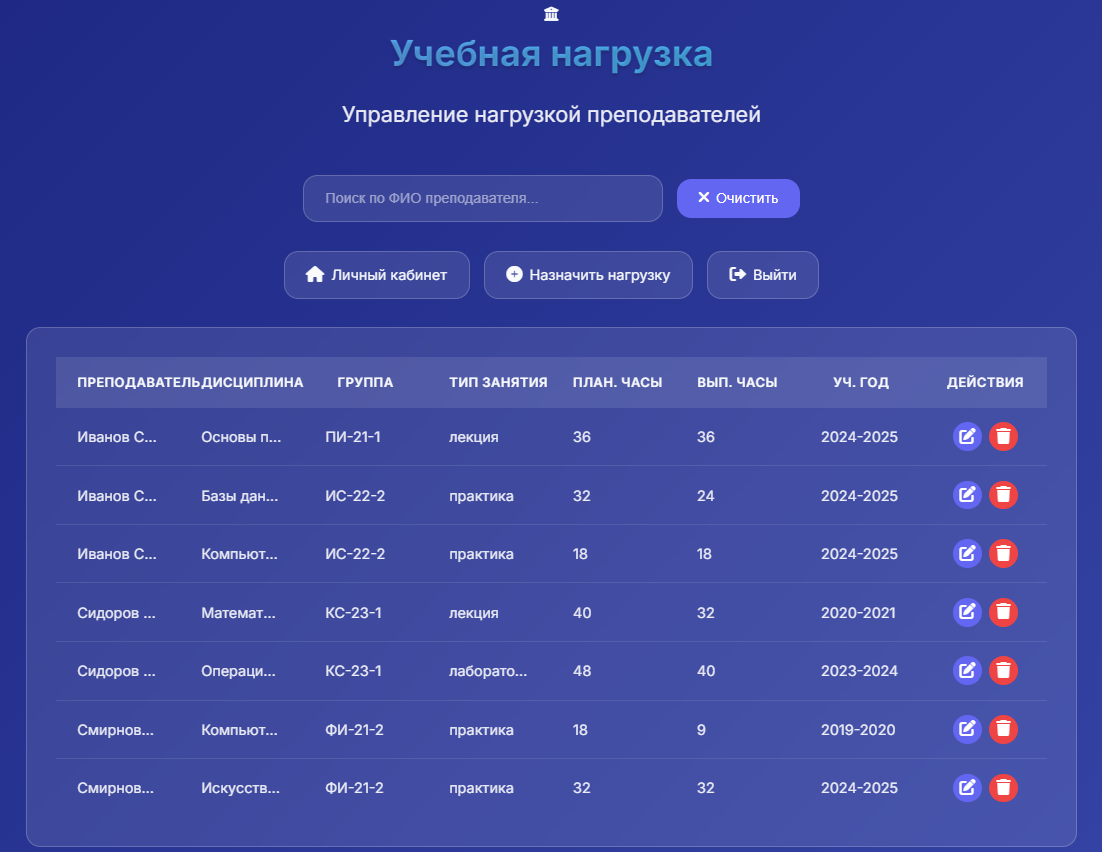
CREATE TABLE `teaching\_loads` (  
 `id` bigint NOT NULL AUTO\_INCREMENT,  
 `teacher\_id` int NOT NULL,  
 `discipline\_id` int NOT NULL,  
 `group\_id` int NOT NULL,  
 `hours` int NOT NULL,  
 `type` enum('лекция','практика','лабораторная') NOT NULL,  
 `hours\_completed` int NOT NULL DEFAULT '0',  
 `academic\_year` varchar(255) DEFAULT NULL,  
 PRIMARY KEY (`id`),  
 KEY `teacher\_id` (`teacher\_id`),  
 KEY `discipline\_id` (`discipline\_id`),  
 KEY `group\_id` (`group\_id`),  
 CONSTRAINT `teaching\_loads\_ibfk\_1` FOREIGN KEY (`teacher\_id`) REFERENCES `teachers` (`id`) ON DELETE CASCADE,  
 CONSTRAINT `teaching\_loads\_ibfk\_2` FOREIGN KEY (`discipline\_id`) REFERENCES `disciplines` (`id`) ON DELETE CASCADE,  
 CONSTRAINT `teaching\_loads\_ibfk\_3` FOREIGN KEY (`group\_id`) REFERENCES `student\_groups` (`id`) ON DELETE CASCADE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

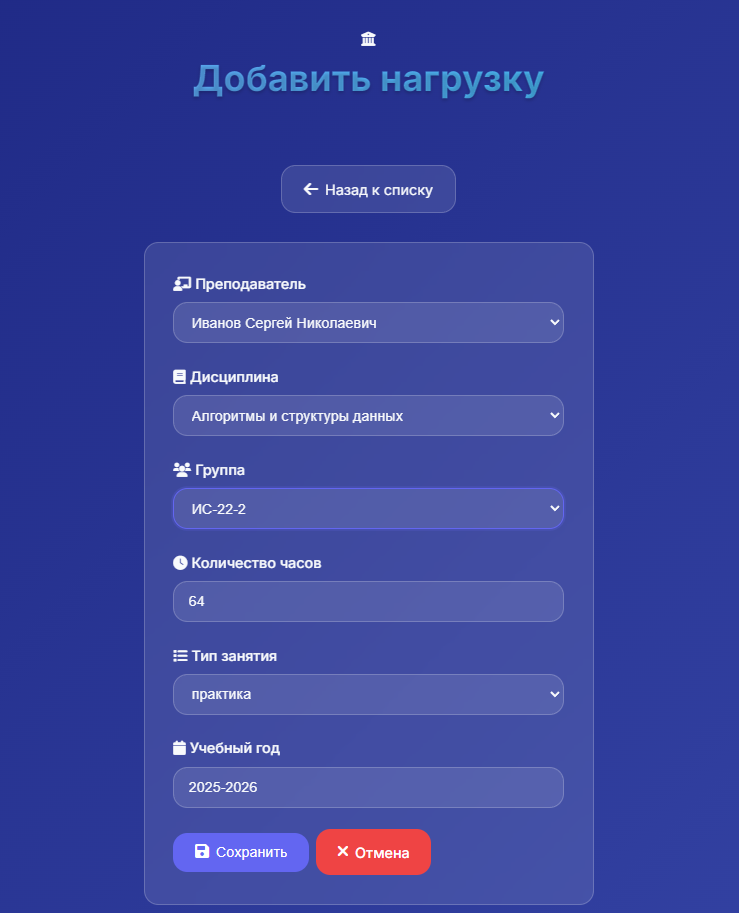
Таблица teaching\_loads хранит информацию о нагрузке преподавателей, ссылаясь на преподавателя, дисциплину и группу.

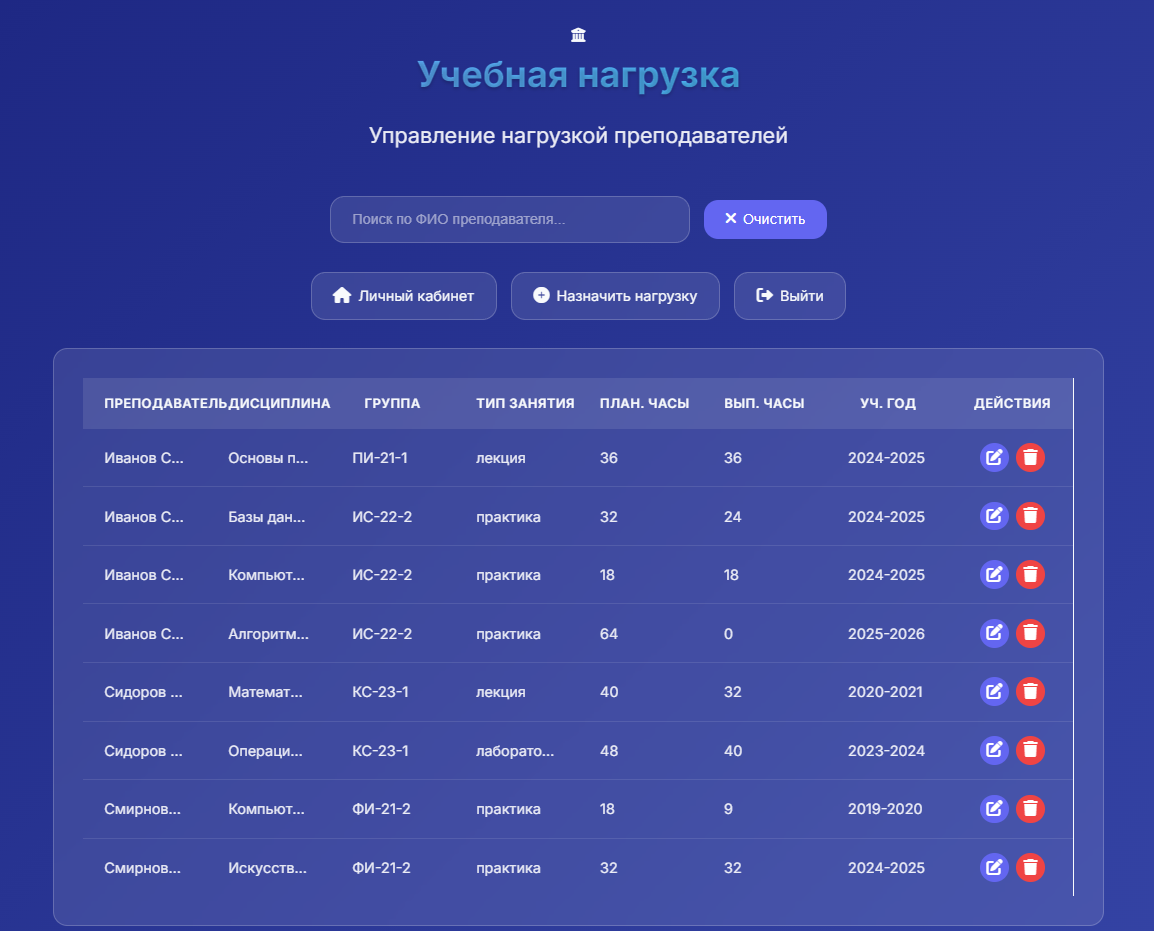
Скриншоты приложения:

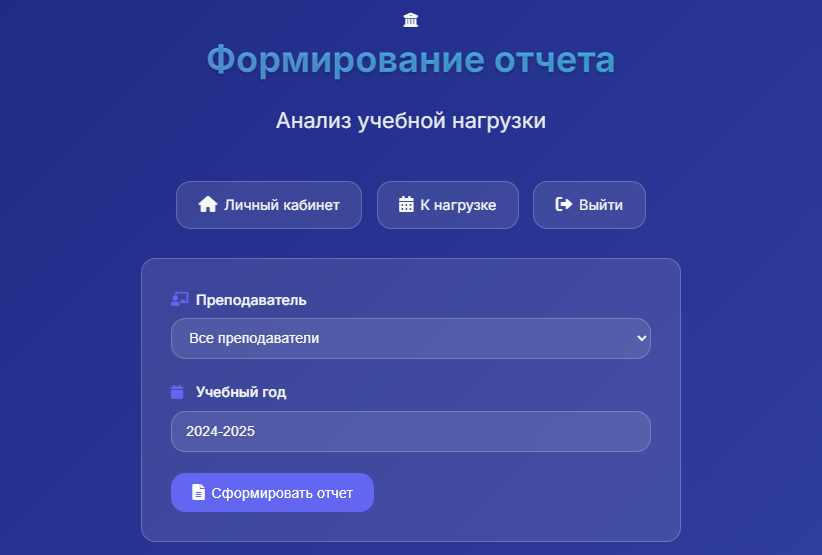
Рисунок 9. Авторизация пользователя.

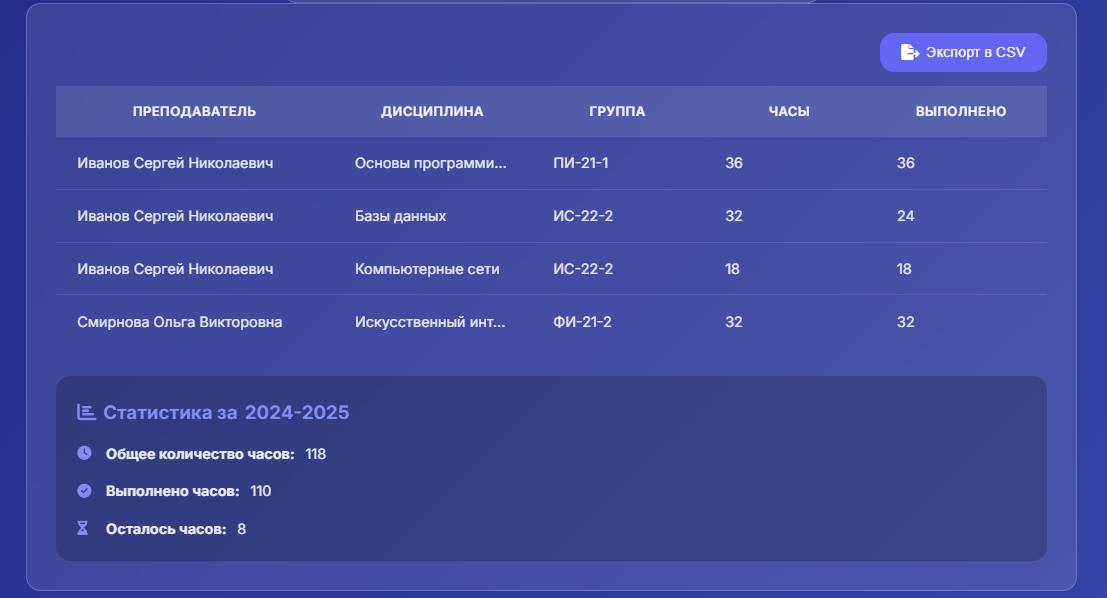
Рисунок 10. Личный кабинет пользователя (заведующего кафедрой).

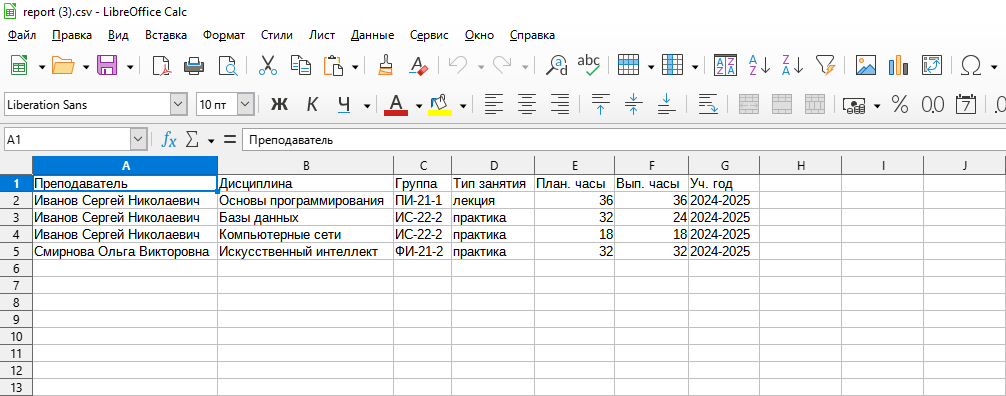
Рисунок 11. Учебная нагрузка которую видит пользователь (заведующий кафедрой).

Рисунок 12. Назначение учебной нагрузки преподавателю.

Рисунок 13. Учебная нагрузка после добавления.

Рисунок 14. Страница для формирования отчёта.

Рисунок 15. Сформированный отчёт.

Рисунок 16. Экспортированный отчёт сформированный заведующим кафедры.

В ходе выполнения курсовой работы была разработана и реализована онлайн-система управления учебной нагрузкой преподавателей. В процессе работы были проанализированы существующие проблемы ручного учета и распределения нагрузки, спроектирована архитектура приложения, разработана структура базы данных и реализован удобный пользовательский интерфейс.