# **ВВЕДЕНИЕ**

В современном мире организация образовательного процесса требует применения инновационных технологий и подходов. В колледжах очень часто возникает проблема с выделением определенных кабинетов для занятий. Разработка такого приложения позволит преподавателям оставлять заявку на занятие нужного кабинета на определенную дату и время, а диспетчеры уже одобрят или отклонят эту заявку. Кроме того, в приложении будут расписание всех занятых кабинетов и описание каждой из них. Приложение поможет университетам и колледжам значительно сократить время на организацию занятий и сделать процесс обучения более эффективным и удобным.

На данный момент в сфере управления аудиторными помещениями в образовательных учреждениях существуют несколько значительных противоречий, которые затрудняют эффективное использование ресурсов:

* ручное распределение кабинетов и координация действий преподавателей значительно усложняют процесс планирования и могут приводить к конфликтам в расписании;
* текущие методы управления занятостью кабинетов затрудняют точное отслеживание свободных пространств и их использования, тогда как автоматизация этого процесса через наше приложение значительно упрощает задачу;
* традиционное управление аудиторным фондом требует дополнительных ресурсов, таких как администраторы для обработки заявок и контроля расписаний, что увеличивает затраты и снижает общую эффективность работы учебного заведения.

Создание приложения для управления аудиторными помещениями в образовательных учреждениях является крайне актуальным шагом, так как она упростит и ускорит процесс планирования и распределения учебных помещений, снизит вероятность ошибок при учете и бронировании кабинетов, а также повысит удобство отслеживания занятости как для преподавателей, так и для диспетчеров. Данная система позволит централизовать информацию о свободных и занятых аудиториях, что, в свою очередь, улучшит координацию и повысит общую эффективность работы учебного заведения.

Использование приложения учета занятости кабинетов даст возможность организациям оптимизировать свои процессы благодаря интеграции с внутренними системами расписания и учета. Это обеспечит эффективное отслеживание статусов занятости помещений, а также необходимых ресурсов для проведения занятий. В результате, управление занятостью кабинетов станет более рациональным, что ускорит процессы взаимодействия и повысит качество предоставляемых образовательных услуг. Преподаватели смогут сосредоточиться на учебном процессе, а не на рутинных задачах, что в конечном итоге повысит общую эффективность работы и удовлетворенность как преподавателей, так и студентов.

При разработке платформы для управления аудиторными помещениями в образовательных учреждениях следует выделить несколько ключевых проблем, которые необходимо учитывать для успешного внедрения и функционирования системы:

* понимание потребностей пользователей: какие функции и возможности преподаватели и диспетчеры хотели бы видеть в новой платформе? Как можно улучшить процесс бронирования и учета кабинетов?
* разработка интуитивно понятного интерфейса: как создать удобный интерфейс, доступный для пользователей с разным уровнем технической компетенции? какие элементы дизайна и навигации будут наиболее эффективными для упрощения взаимодействия в условиях учебного процесса?
* интеграция с существующими системами управления: как обеспечить бесшовную интеграцию с текущими системами учета и расписания, минимизируя при этом возможные сбои в работе?
* управление данными и безопасность: какие меры безопасности необходимо внедрить для гарантии конфиденциальности и защиты информации о учебных процессах и пользователях платформы?
* тестирование и отладка: какие тесты необходимо выполнить на различных этапах разработки платформы? Какие будут сценарии использования приложения в реальных условиях учебного заведения?
* адаптивность платформы для различных устройств (мобильные телефоны, планшеты, ПК) и масштабируемость для будущих расширений функционала: как заложить такие характеристики в архитектуре приложения, чтобы обеспечить максимальную эффективность и доступность для всех пользователей?

Каждый из этих аспектов требует детального исследования и тщательной проработки для успешного внедрения приложения для учета занятости кабинетов, чтобы обеспечить его эффективность, удобство и безопасность.

Объект исследования – процесс разработки приложения для оптимизации управления занятостью кабинетов в колледже.

Предмет исследования – программное обеспечение учет занятости кабинетов, предназначенный для автоматизации процесса бронирования кабинетов преподавателями.

Цель курсового проекта: разработать функциональный и удобный web-сайт, который позволит преподавателям эффективно управлять занятостью кабинетов и упрощать процесс подачи заявок на бронирование.

Задачи:

* провести исследование для определения потребностей и предпочтений целевой аудитории (как преподавателей, так и диспетчеров);
* спроектировать структуру и функциональность приложения, выбрать технологии и инструменты для разработки;
* реализовать разработанный web-сайт с интуитивно понятным интерфейсом.

Гипотеза исследования: автоматизация процесса бронирования кабинетов с помощью приложения для учета занятости кабинетов с удобным интерфейсом позволит преподавателям быстро и просто оставлять заявки на занятие кабинетов, сократит время на ожидание одобрения и повысит общую эффективность организации учебного процесса.

Для изучения проблем разработки приложения учета занятости кабинетов будут применяться следующие методы исследования:

* опросы и анкетирование: проведение опросов среди преподавателей и диспетчеров колледжа с целью сбора информации о их потребностях, предпочтениях и текущих проблемах в процессе бронирования кабинетов;
* глубинные интервью с экспертами: получение более глубокого понимания опыта и ожиданий пользователей через интервью с преподавателями и административным персоналом, работающим с занятостью кабинетов;
* анализ конкурентов: изучение существующих решений для управления занятостью кабинетов, анализ их функциональности и пользовательского интерфейса для выявления лучших практик и возможных недостатков;
* пользовательское тестирование: создание прототипов приложения и их тестирование на целевой аудитории для получения обратной связи о удобстве использования и функциональности на ранних этапах разработки;
* моделирование сценариев использования: разработка сценариев, описывающих разные пути взаимодействия пользователей с приложением, что позволит выявить возможные проблемы и улучшить интерфейс;
* кросс-дисциплинарные исследования: вовлечение специалистов из различных областей (дизайнеров, разработчиков, методистов) для обеспечения комплексного подхода к разработке приложения;
* объектно-ориентированный анализ: проектирование архитектуры и функциональности приложения, построение моделей поведения системы;
* анализ современного инструментария для разработки программного обеспечения: изучение актуальных инструментов и технологий, которые могут быть использованы при создании приложения;
* методы математической статистики: использование статистических методов для анализа данных с целью выявления закономерностей в потребностях пользователей;
* алгоритмы применения инструментальных средств: разработка алгоритмов и методик применения инструментальных средств на различных этапах разработки приложения.

Практическая значимость курсового проекта заключается в возможности использования приложения для учета занятости кабинетов в учебном процессе, что значительно повысит эффективность управления распределением кабинетов как для преподавателей, так и для административного персонала.

Основными пользователями разработанного приложения станут преподаватели и диспетчеры, которые занимаются организацией учебного процесса в колледже.

Аспекты проектирования приложения будут рассмотрены в первом разделе дипломного проекта.

Результаты практической реализации спроектированного приложения будут представлены во втором разделе дипломного проекта.

# **1 ТЕХНИЧЕСКИЙ ПРОЕКТ**

## Анализ предметной области

В современном образовательном процессе эффективное управление ресурсами, такими как учебные аудитории, становится важным аспектом для обеспечения качественного обучения. В колледже, где существует множество дисциплин и преподавателей, необходимость в оптимизации использования учебных кабинетов становится особенно актуальной. Преподаватели часто сталкиваются с проблемой недостатка информации о доступности кабинетов, что может привести к конфликтам и неэффективному использованию ресурсов.

Приложение учет занятости кабинетов позволяет решить эту проблему, предоставляя пользователям удобный интерфейс для просмотра расписания, подачи заявок на использование кабинетов и отслеживания статуса этих заявок. Аудитории, как ключевые ресурсы образовательного процесса, должны быть доступны для преподавателей в соответствии с их потребностями и расписанием занятий.

Некоторые из этих преимуществ включают:

* доступность и удобство использования – web-сайт доступен для преподавателей и диспетчеров в любое время и на любом устройстве. Он имеет интуитивно понятный интерфейс и простую навигацию, что облегчает процесс бронирования кабинетов;
* поиск по расписанию и заявкам – удобная функция поиска позволяет быстро находить нужные расписания занятий или заявки, что экономит время и улучшает доступ к информации;
* широкая аудитория – web-сайт будет полезен для всех преподавателей и диспетчеров колледжа, независимо от их уровня технической подготовки. Это делает его универсальным инструментом для упрощения процесса распределения кабинетов.  
    
  Приложение призвано решать ряд актуальных задач, таких как:
* упрощение коммуникации – обеспечивает быструю и прямую связь между преподавателями и диспетчерами, позволяя обмениваться сообщениями и оставлять заявки на аудитории;
* сокращение времени на поиск информации – пользователи могут быстро находить необходимые расписания занятий и информацию о кабинетах, а также отслеживать статус своих заявок;
* организация использования кабинетов – поддерживает систему бронирования кабинетов, что улучшает взаимодействие между преподавателями и диспетчерами;
* улучшение корпоративной культуры – способствует более прозрачному и доступному взаимодействию между преподавателями и административным персоналом, что укрепляет командный дух и повышает уровень вовлеченности;
* обратная связь и одобрение заявок – позволяет диспетчерам быстро реагировать на заявки преподавателей, обеспечивая более эффективное распределение ресурсов;
* поддержка удаленной работы и гибкости – обеспечивает связь и доступ к информации для преподавателей в различных условиях работы, что делает процесс бронирования более удобным для всех участников.

## Постановка задачи

Необходимо разработать web-приложение для образовательного учреждения.

Приложение должно будет содержать все необходимые сведения об образовательном учреждении, иметь удобный и интуитивно понятный пользователю графический интерфейс

Приложение необходимо разработать с использованием трехуровневой архитектуры. Для бизнес-логики необходимо будет разработать API.

Web-приложение должна обеспечивать выполнение следующих функций:

* авторизация пользователей, имеющих доступ к платформе;
* предоставление в главном окне основной информации: расписание пар, а также занятость кабинетов;
* возможность загружать расписание пар;
* возможность преподавателю зайти в свой профиль;
* возможность подачи заявки диспетчеру на бронирование кабинета на определенную пару и дату;
* отображение списка кабинетов с их описанием, включая характеристики (вместимость, наличие мультимедийного оборудования и т.д.);
* возможность отслеживания статуса поданных заявок (на рассмотрении, одобрена, отклонена) в специальной вкладке «Заявки»;
* функция фильтрации и поиска по аудиториям для удобства выбора подходящего кабинета;
* хранение материалов в БД.

Программный продукт должен включать в себя:

* API для бизнес-логики;
* дизайнерскую часть, которая обеспечит удобную работу с web-сайтом, обеспечит удобный и интуитивно понятный интерфейс для навигации по нему;
* web-приложение для управления портала.

Требования по входным данным:

* запросы пользователей – отправка заявки на бронирование кабинета;
* данные диспетчера – включает информацию, необходимую для управления данными приложения.
* данные аутентификации и учетной записи – включают информацию, предоставляемую администраторами входе в приложение. Эти данные включают идентификационную информацию, такую как логин, имя пользователя и пароль.

Требования по выходным данным:

* заявки пользователей – выходные данные представляют собой заявки, отправленные преподавателями на использование кабинетов. Эти данные могут содержать информацию о дате, времени и названии кабинета;
* результаты поиска – включают результаты поиска по расписанию пар и списку кабинетов, которые соответствуют запросам пользователей. Эти данные содержат информацию о доступных аудиториях и их расписании;
* интерфейс диспетчера – выходные данные для диспетчеров включают интерфейс управления заявками и расписанием, который позволяет им одобрять или отклонять заявки, а также добавлять и редактировать информацию о кабинетах;
* данные ответов на запросы от API – предоставляют внешним приложениям доступ к информации о расписании занятий и статусе заявок. Эти данные должны быть в формате JSON и могут содержать информацию о свободных и занятых кабинетах;
* данные учетных записей – включают информацию о профилях преподавателей и диспетчеров, их настройках и правах доступа;
* уведомления и сообщения – включают уведомления для пользователей о статусе их заявок, подтверждении бронирования кабинетов, изменениях в расписании и ошибках системы.

Запросы должны обрабатываться через API и передаваться внешним приложениям в формате JSON.

Хранение и передача данных должны осуществляться в безопасной и защищенной форме. Доступ к данным для администраторов должен быть строго контролируемым с использованием аутентификации. Информация администраторов должна храниться в зашифрованном виде, и необходимо принять меры для обеспечения их конфиденциальности, чтобы предотвратить несанкционированный доступ к правам администратора.

**1.3 Проектирование функциональности web-приложения**

Для описания взаимодействия преподавателей и диспетчера с приложением «Учет занятости кабинетов» воспользуемся методологией UML.

Рассмотрим на диаграмме вариантов использования взаимодействие пользователей с разрабатываемым приложением. Для этого построим диаграмму Use Case, в которой имеются:

* актеры («Преподаватель», «Диспетчер», «API», «DB»);
* прецеденты («Авторизация», «Отправка заявки», «Получение заявки», «Одобрение заявки», «Отклонение заявки», «Обновление информации о пользователе», «Обработка запроса API», «Обработка запроса БД»).

Диаграмма вариантов использования представлена в приложении А.

Рассмотрим ключевые моменты диаграммы:

* Пользователь может выполнять прецеденты «Авторизация», «Просмотр расписания», «Отправлять заявки для занятия кабинета», «Просмотр списка кабинетов»;
* «Администратор» может выполнять прецеденты «Загружать расписание», «Одобрять/отклонять заявки» и также имеет доступ к прецедентам «Авторизация», «Просмотр расписания», «Просмотр списка кабинетов»;
* API обрабатывает запросы от актеров и взаимодействует с БД;
* DB обрабатывает запросы от API и доступ к данным.

Все прецеденты выполняются после прецедента «Авторизация», что обеспечивает контроль доступа к управлению участниками чатов.

Прецедент «Обработка запроса API» проверяет роль пользователя, а затем выполняет запрос. «Обработка запроса API» включает в себя прецедент «Обработка запроса БД», который выполняется при выполнении условий «Обработка запроса API» и получает необходимые данные из базы данных для удовлетворения запроса пользователя.

**1.4 Проектирование структуры программного продукта**

С целью удостоверения предъявляемых к программе требований, приведенных в разделах 1.1-1.2, при реализации портала будет использоваться трехзвенная архитектура. Данная архитектура представляет собой организацию приложения на трех уровнях, каждый из которых выполняет определенные функции:

* представление этот уровень отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем. Он включает компоненты, которые занимаются визуализацией данных и обработкой действий пользователей. В этот уровень будет входить веб-приложение;
* логика приложения – на этом уровне осуществляется обработка бизнес-логики, связанной с функциональностью системы управления кабинетами. Здесь располагаются компоненты, отвечающие за обработку пользовательских запросов, проверку прав доступа, обработку данных и взаимодействие с базой данных;
* слой данных – уровень отвечает за хранение и доступ к данным. Здесь находится БД, содержащая информацию, и компоненты, реализующие операции чтения и записи данных. На портале этот слой будет представлять СУБД и компоненты, работающие с ней для получения и записи данных.

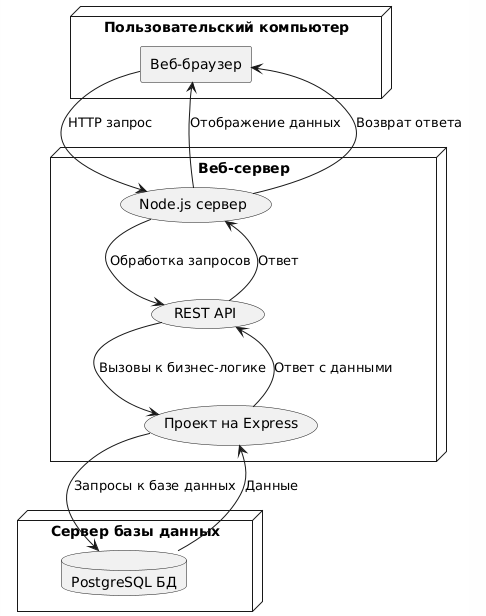


Рисунок 1 – Схема развёртывания

Разрабатываемое web-приложение будет состоять из следующих составляющих:

* аутентификация и авторизация – форма входа для аутентификации администраторов с разделением прав доступа;
* главная страница – центральная страница, на которой пользователи могут просматривать расписание пар и занятость аудиторий;
* страница список кабинетов – на этой странице находится список кабинетов с их характеристиками;
* страница занять кабинет – страница, на которой можно отправлять заявку на бронирование кабинета;
* страница заявок – страница, на которой преподаватели будут видеть свои заявки, а также их состояние (одобрено, отклонено, на рассмотрении), а диспетчер на этой странице может одобрять или отклонять эти заявки;
* страница пользователя – на этой странице можно изменять некоторые данные о пользователе.

Тестирование программ планируется проводить путём выбора каждого пункта меню и проверки корректности получаемого отклика от программы. При тестировании будет производиться эмуляция различных вариантов пользовательских действий.

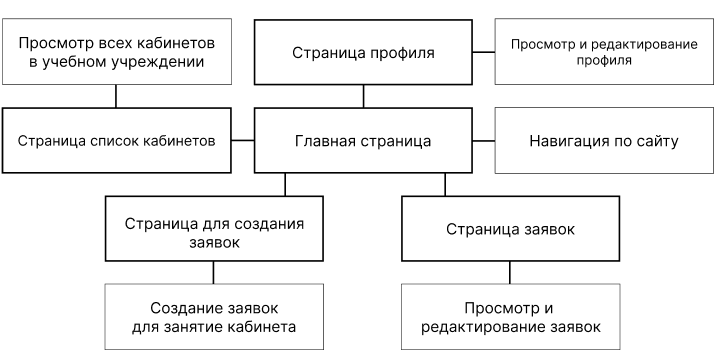
****

Рисунок 2 – Функциональные модули web-приложения

**1.5 Проектирование базы данных**

Для хранения данных о сообщениях будет разработана БД. Она будет состоять из следующих таблиц:

* Пользователь (user\_id, password, surname, name, lastname, role\_id);
* Роль (role\_id, role\_name);
* Пара (lesson\_id, user\_id, lesson\_number, subject\_id, date);
* Аудитория (auditorium\_id, number, description, computers, interactive\_whiteboard);
* Предмет (subject\_id, name, description);
* Запрос (requests \_id, user\_id, auditorium\_id, class\_id, request\_date, status, message);
* Преподаватели и предметы (user\_id, subject\_id);

Логическая модель БД представлена в приложении Б.

Ниже приведён словарь данных:

Таблица 1 – Таблица «Пользователь» (User)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| user\_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| login | varchar(255) |  | Нет | UNIQUE |
| password | varchar(255) |  | Нет |  |
| surname | varchar(50) |  | Нет |  |
| name | varchar(50) |  | Нет |  |
| lastname | varchar(50) |  | Да |  |
| role\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Роль.role\_id) |

Таблица 2 – Таблица «Роль» (Role)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| role\_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| role\_name | varchar(255) |  | Нет | UNIQUE |

Таблица 3 – Таблица «Учебное занятие» (Lesson)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| lesson\_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| user\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Пользователь.user\_id) |
| lesson\_number | int(11) | UNSIGNED | Нет |  |
| subject\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Предмет.subject\_id) |
| auditorium\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Аудитория.auditorium\_id) |
| date | dateitme |  | Нет |  |

Таблица 4 – Таблица «Кабинет» (Auditorium)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| auditoriums \_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| number | int(11) | UNSIGNED | Нет | UNIQUE |
| description | varchar(255) |  | Да |  |
| number\_of\_seats | int(11) | UNSIGNED | Нет |  |
| computers | boolean |  | Нет |  |
| interactive\_whiteboard | boolean |  | Нет |  |

Таблица 5 – Таблица «Предмет» (Subject)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| subject\_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| name | varchar(255) |  | Нет | UNIQUE |
| description | varchar(255) |  | Нет |  |

Таблица 6 – Таблица «Заявка» (Request)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| request\_id | bigint(20) | UNSIGNED | Нет | auto\_increment |
| user\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Пользователь.user\_id) |
| lesson\_number | int(11) | UNSIGNED | Нет |  |
| subject\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Предмет.subject\_id) |
| request\_date | dateitme |  | Нет |  |
| auditorium\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Аудитория.audirorium\_id) |
| status | varchar(255) |  | Нет |  |
| message | varchar(255) |  | Да |  |

Таблица 7 – Таблица «Преподаватели и предметы» (Teachers\_and\_subjects)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип** | **Атрибуты** | **Null** | **Дополнительно** |
| subject\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Предмет.subject\_id) |
| user\_id | bigint(20) | UNSIGNED | Нет | FK (ссылается на Пользователь.user\_id) |

**1.6 Проектирование пользовательского интерфейса**

Проектирование пользовательского интерфейса для учета занятости кабинетов – это важный шаг, который требует учета различных аспектов. Ниже приведу схемы спроектированного пользовательского интерфейса:

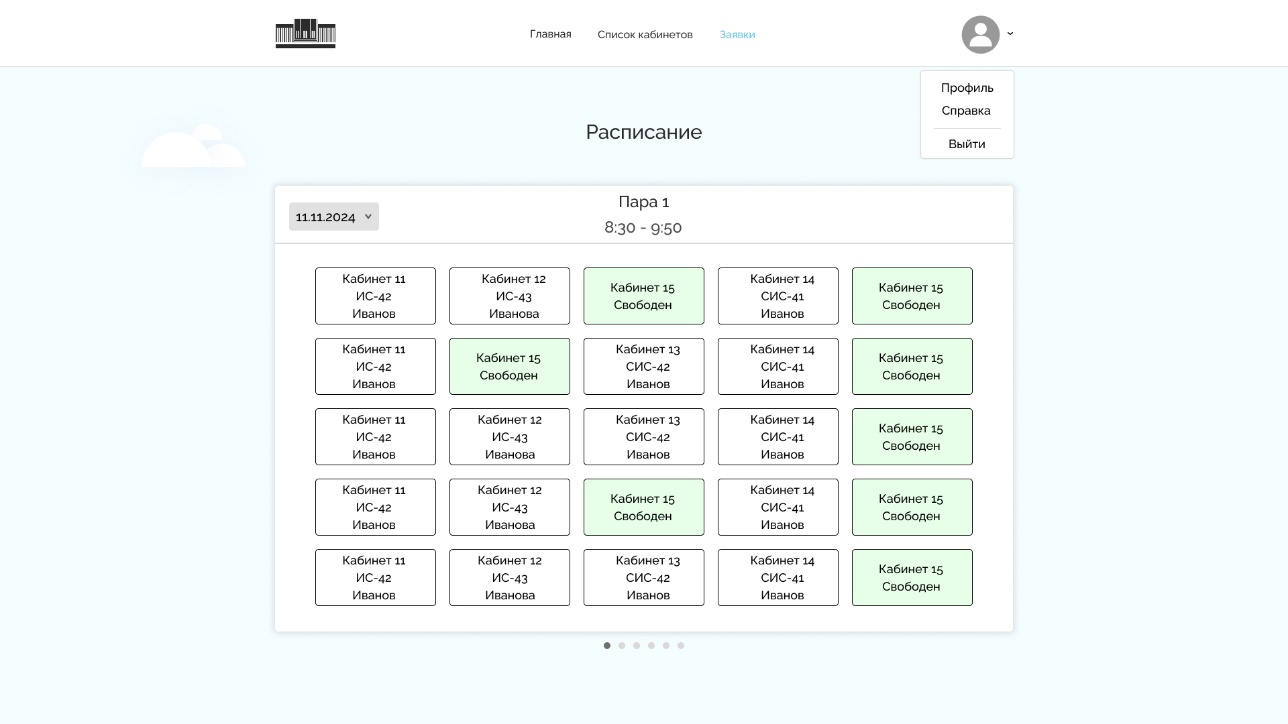


Рисунок 3 – Главная страница

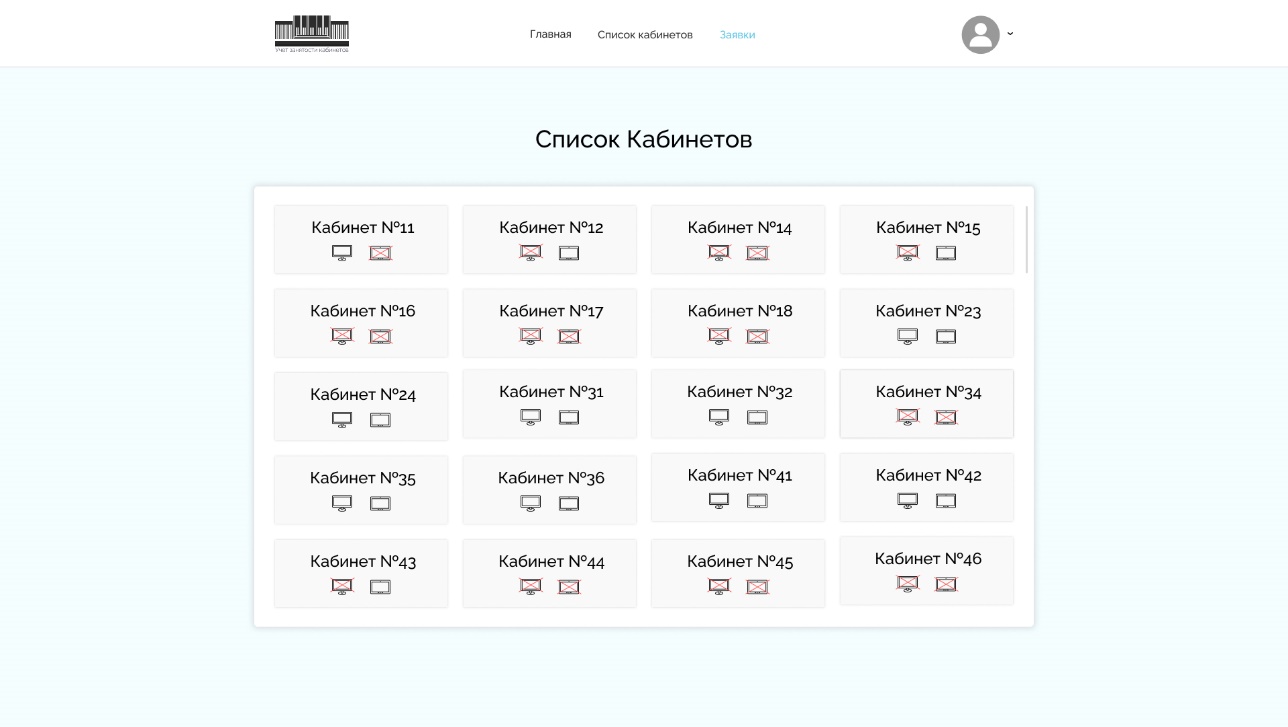


Рисунок 4 – Страница список кабинетов

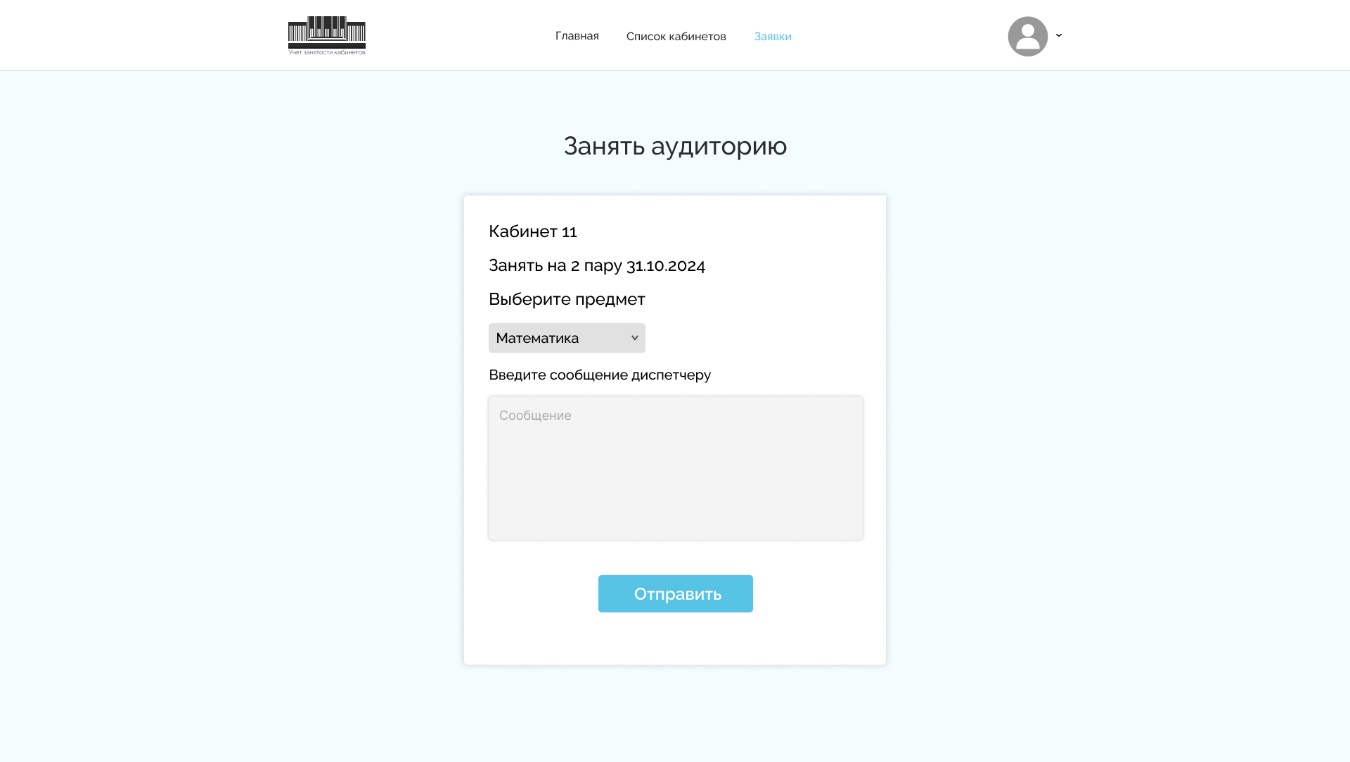


Рисунок 5 – Страница для создания заявок

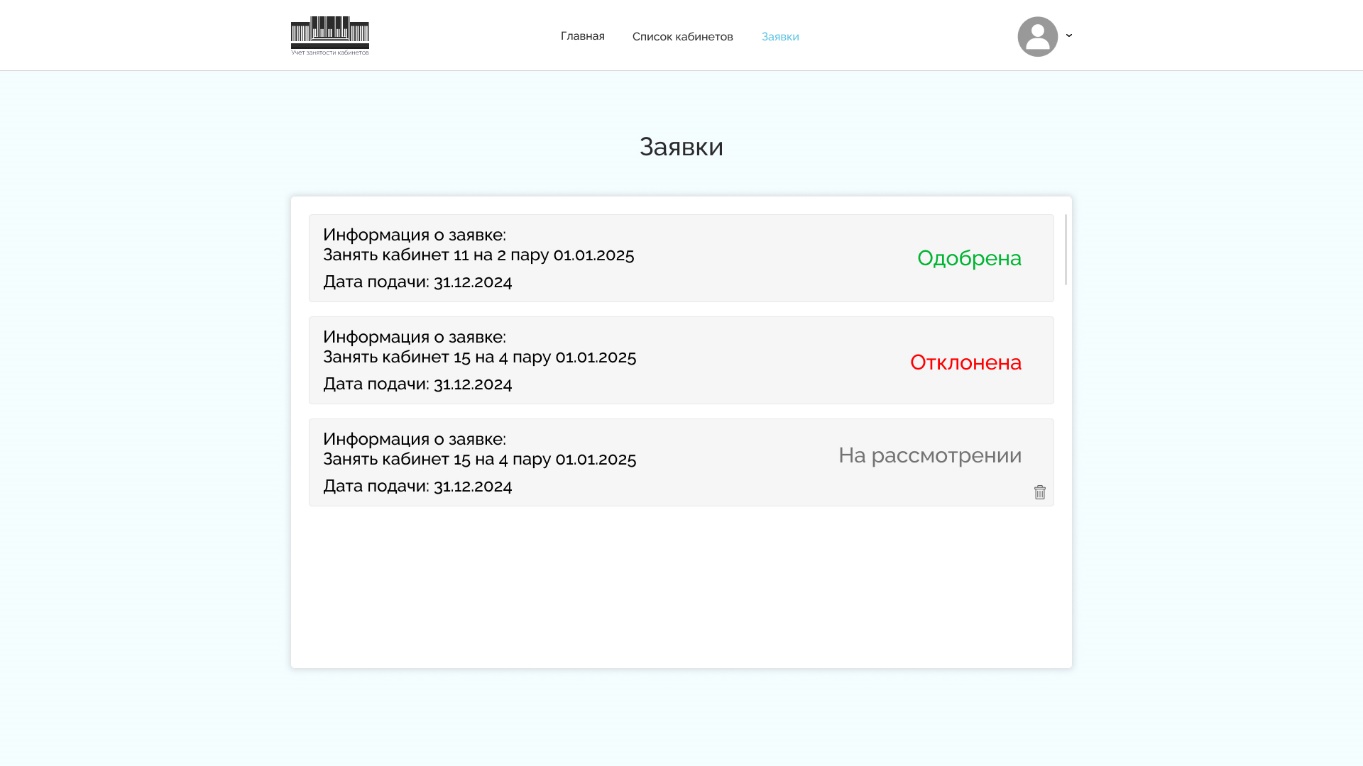


Рисунок 6 – Страница заявок

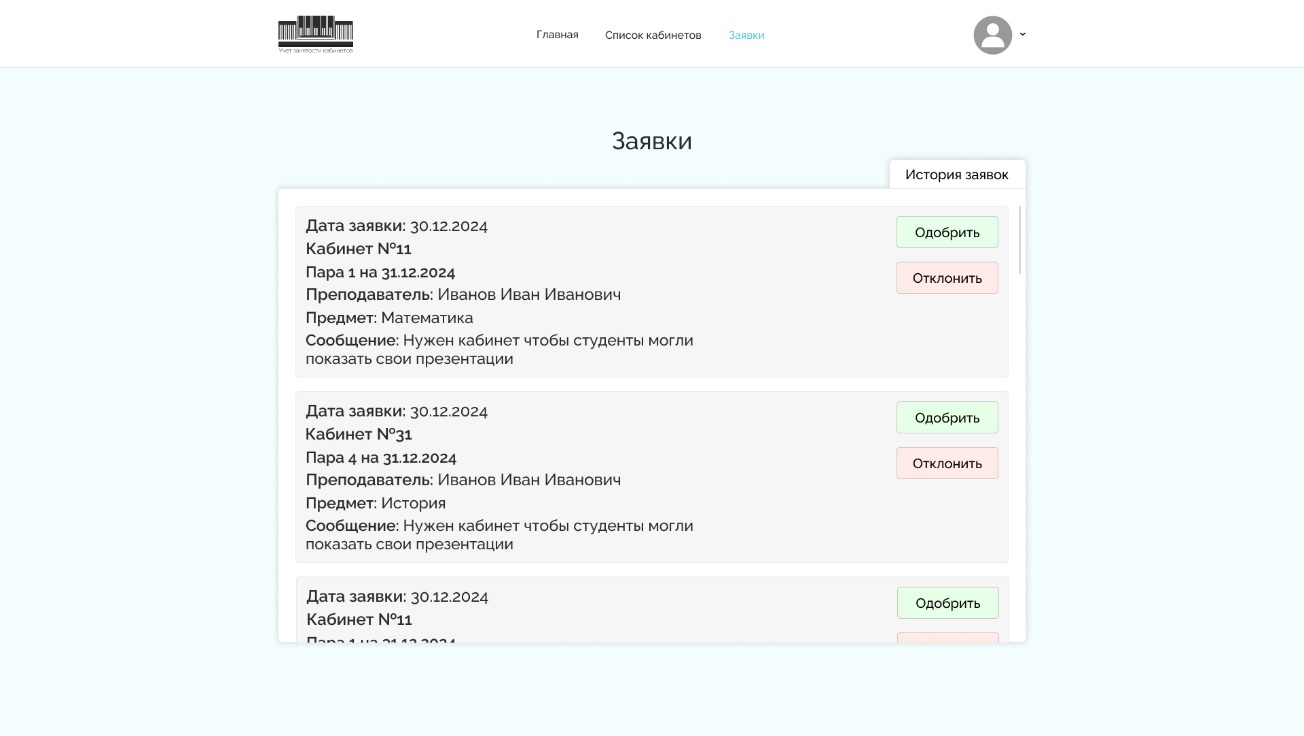


Рисунок 7 – Страница заявок для диспетчера

Ниже приведены схемы пользовательских путей, на основные пути:

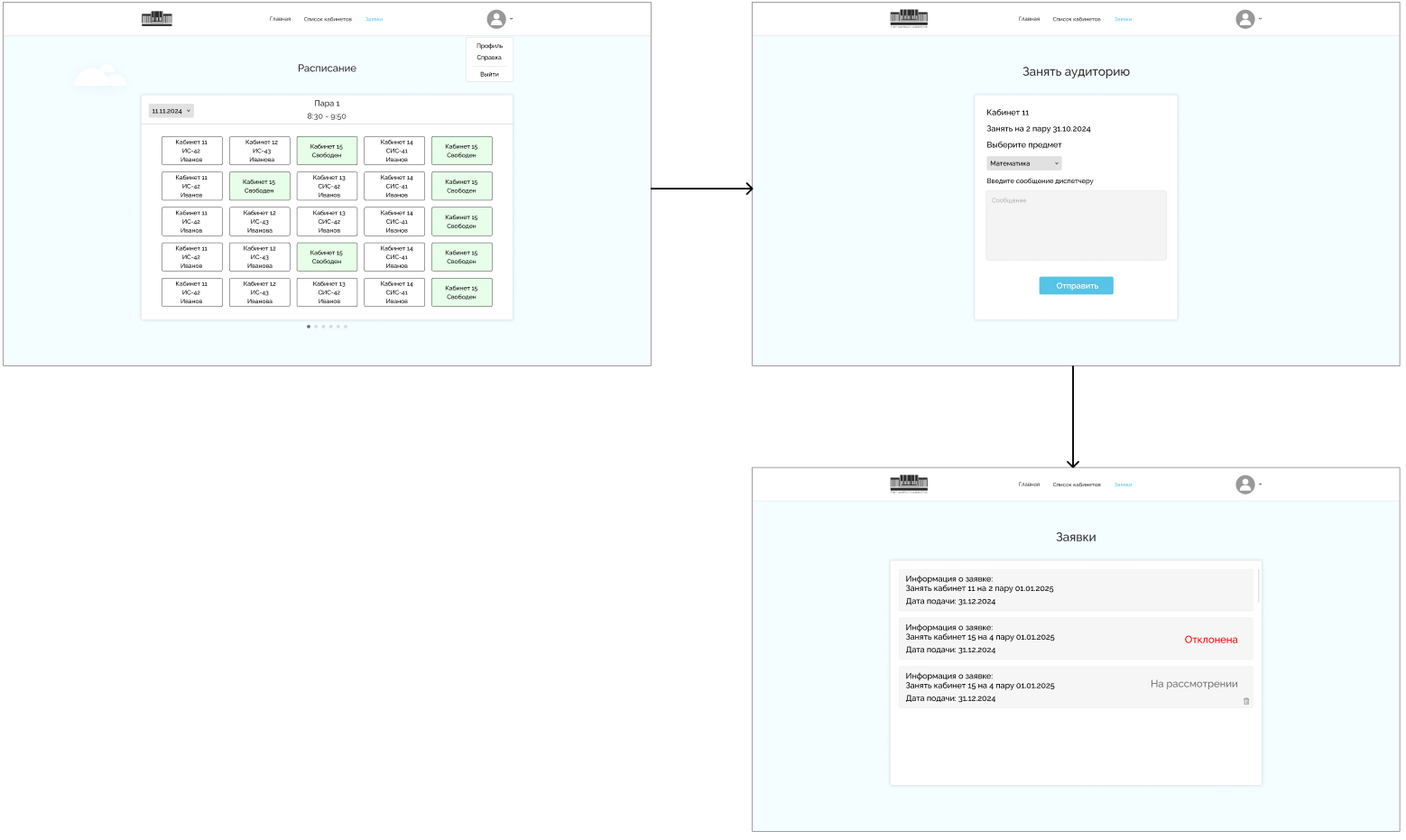


Рисунок 8 – Схема пользовательских путей для прецедента «Создания заявки бронирование кабинета»

**1.7 Выбор средств разработки**

Среди различных языков программирования для курсового проекта был выбран JavaScript, а для разработки использовалась интегрированная среда Visual Studio Code (1.95). Для разработки клиентской части web-приложения использовались HTML и CSS в сочетании с фреймворком React.js, в то время как серверная часть разрабатывалась с помощью Node.js Express. Для управления базами данных была выбрана СУБД PostgreSQL.

JavaScript – это высокоуровневый, интерпретируемый язык программирования, который преимущественно используется для создания динамических и интерактивных веб-страниц.

JavaScript относится к динамическим языкам программирования и выполняется на стороне клиента (в браузере пользователя). Он позволяет изменять содержимое веб-страницы без необходимости перезагрузки всей страницы. Также он обеспечивает взаимодействие с пользователем (например, обработку кликов, наведения мыши, отправки форм). JavaScript позволяет проверять данные, введенные в формы, до их отправки на сервер.

Язык HTML (HyperText Markup Language) – это основной язык разметки, используемый для создания веб-страниц. Он предоставляет структуру и семантику контента, позволяя определить, какие элементы на странице являются заголовками, параграфами, ссылками, изображениями и так далее. HTML используется для описания содержимого веб-страницы, а не её визуального представления. Визуальное оформление страницы определяется с помощью CSS (Cascading Style Sheets).

HTML состоит из различных элементов, таких как теги, атрибуты и текстовое содержимое. Теги используются для определения начала и конца элемента, например <p> для параграфа или <img> для изображения. Атрибуты предоставляют дополнительную информацию об элементе, такую как его идентификатор, класс, ссылку на изображение и так далее. Текстовое содержимое вставляется между открывающим и закрывающим тегами элемента.

HTML также поддерживает использование различных типов элементов, таких как формы для сбора информации от пользователей, таблицы для представления данных, списки для структурирования информации и многие другие.

Основной целью HTML является создание структурированного контента, который может быть интерпретирован браузерами и отображен в виде веб-страницы. В современном веб-разработке HTML часто комбинируется с другими технологиями, такими как CSS для стилизации и JavaScript для добавления интерактивности на страницу.

React.js – это библиотека для языка программирования [JavaScript](https://blog.skillfactory.ru/glossary/javascript/) с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность — явление, когда в ответ на изменение одного элемента меняется все остальное. У React открытый исходный код и мощное сообщество. Это одна из самых популярных библиотек для веб-разработки.

Node.js — это прогрессивная среда выполнения JavaScript на стороне сервера, основанная на движке V8 от Google, что позволяет создавать масштабируемые и высокопроизводительные приложения. Благодаря неблокирующей архитектуре и асинхронной модели I/O, Node.js обеспечивает эффективное управление запросами и быстрый отклик серверов, что особенно ценно в условиях обильного трафика.

Express – это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Express значительно упрощает процесс разработки серверной части приложений, предлагая простую, но мощную структуру, которая позволяет быстро настраивать маршруты, обрабатывать запросы и управлять сессиями. Его гибкость и возможность интеграции с различными промежуточными программами делают его идеальным выбором для создания RESTful API.

PostgreSQL будет выступать в роли главной реляционной системы управления базами данных. Она предоставляет богатый функционал для хранения и обработки данных, включая поддержку JSON, возможность транзакций и расширенные функции для запросов. Это создаст надежное и масштабируемое решение для хранения пользовательских данных, сообщений и прочей важной информации.

**1.8 Проектирование тестовых пакетов**

|  |  |
| --- | --- |
| **Название проект** | «Учет занятости кабинетов» |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Тимур |
| **Дата теста** | **12.11.2024** |

Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_BF\_1 (basic functionality test case #1). |
| **Приоритет тестирования** | Средний. |
| **Заголовок/название теста** | Тестирование авторизации с заведомо верными данными. |
| **Краткое изложение теста** | Тестирование авторизации с заведомо верными логином и паролем. |
| **Этапы теста** | 1. Открыть окно авторизации; 2. Ввести в поле логин «testlogin»; 3. Ввести в поле пароль «testpass»; 4. Нажать на кнопку «войти». |
| **Тестовые данные** | Логин: «testlogin»; Пароль «testpass». |
| **Ожидаемый результат** | Окно авторизации закрылось и открылось новое окно, где отображается главная страница программы. |
| **Фактический результат** | Окно авторизации закрылось и открылось новое окно, где отображается главная страница программы. |
| **Статус** | Зачет. |
| **Предварительное условие** | В базе данных имеется запись о пользователе с логином «testlogin» и паролем «testpass». Имеется подключение к базе данных. |
| **Постусловие** | В программе открыта главная страница, в заголовке главной страницы в поле ФИО отображается Иванов Иван Иванович. |
| **Примечания/комментарии** |  |

Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_BF\_2 (basic functionality test case #2). |
| **Приоритет тестирования** | Высокий. |
| **Заголовок/название теста** | Тестирование авторизации с заведомо неверными данными. |
| **Краткое изложение теста** | Тестирование авторизации с заведомо неверными логином и паролем. |
| **Этапы теста** | 1. Открыть окно авторизации; 2. В поле логин ввести «testsfihg»; 3. В поле пароль ввести «testsejgnj»; 4. Нажать на кнопку «войти». |
| **Тестовые данные** | Логин: «testsfihg»; Пароль «testsejgnsdj». |
| **Ожидаемый результат** | Под полем для ввода пароля отображается красный текст «Логин/пароль введены неверно». |
| **Фактический результат** | Под полем для ввода пароля отображается красный текст «Логин/пароль введены неверно». |
| **Статус** | Зачет. |
| **Предварительное условие** | TC\_BF\_2 (basic functionality test case #2). |
| **Постусловие** | Высокий. |
| **Примечания/комментарии** | Тестирование авторизации с заведомо неверными данными. |

Тестовый пример #3:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_ER\_1 (error handling test case #1). |
| **Приоритет тестирования** | Высокий. |
| **Заголовок/название теста** | Тестирование авторизации при отсутствии подключения к базе данных. |
| **Краткое изложение теста** | Тестирование авторизации при отсутствии подключения к базе данных |
| **Этапы теста** | 1. Открыть окно авторизации; 2. В поле логин ввести «testlogin»; 3. В поле пароль ввести «testpass»; 4. Нажать на кнопку «войти». |
| **Тестовые данные** | Логин: «testlogin»; Пароль «testpass». |
| **Ожидаемый результат** | Отображается окно с ошибкой – «Ответ не получен. Проверьте подключение и попробуйте еще раз». |
| **Фактический результат** | Программа прекращает работу с необработанной ошибкой. |
| **Статус** | Незачет. |
| **Предварительное условие** | Отсутствует подключение к базе данных. |
| **Постусловие** | Поверх окна с авторизацией отображается окно с ошибкой. |
| **Примечания/комментарии** |  |

Тестовый пример #4:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_BF\_3 (basic functionality test case #3). |
| **Приоритет тестирования** | Средний. |
| **Заголовок/название теста** | Добавление клиента с некорректными данными. |
| **Краткое изложение теста** | Тестирование на добавления пользователя с отсутствующей фамилией. |
| **Этапы теста** | 1. На вкладке добавления клиента поле «фамилия» оставить пустым; 2. На вкладке добавления клиента в поле «имя ввести» «Иван»; 3. На вкладке добавления клиента в поле «отчество» ввести «Иванович»; 4. На вкладке добавления пользователя в поле пароль «Password»; 5. На вкладке добавления пользователя нажать на кнопку «Добавить пользователя»; |
| **Тестовые данные** | * Имя «Иван»; * Отчество «Иванович»; * пароль «Password»; |
| **Ожидаемый результат** | Отображается всплывающее окно с текстом «Поле фамилия не может быть пустым!». |
| **Фактический результат** | Отображается всплывающее окно с текстом «Поле фамилия не может быть пустым!». |
| **Статус** | Зачет. |
| **Предварительное условие** | Вкладка добавление пользователя активна. Имеется подключение к базе данных. В базе данных не имеется записи о пользователе с паролем «Password». |
| **Постусловие** | Запись о пользователе не была добавлена в базу данных. |
| **Примечания/комментарии** |  |

Тестовый пример #5:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_BS\_4 (basic functionality test case #4). |
| **Приоритет тестирования** | Низкий. |
| **Заголовок/название теста** | Скрытие и показ пароля. |
| **Краткое изложение теста** | Тестирование показа и скрытия пароля. |
| **Этапы теста** | 1. На окне авторизации в поле «пароль» ввести «1234»; 2. Поставить галочку в «показать пароль»; 3. Убрать галочку с «показать пароль». |
| **Тестовые данные** | * Пароль «1234». |

**1.9 Требования к программному обеспечению и техническим средствам**

Для корректной работы web-сайта необходимы следующие требования:

* процессор:
* минимальные требования – x86- или x64-разрядный двухъядерный процессор с тактовой частотой 1,7 ГГц, с набором инструкций SSE2;
* рекомендуемые требования – x64-разрядный двухъядерный процессор с тактовой частотой 3 ГГц и выше, с набором инструкций SSE2;
* память:
* минимальные требования – ОЗУ объемом 2 Гб;
* рекомендуемые требования – ОЗУ объемом 4 Гб и более;
* дисплей:
* минимальные требования – VGA с разрешением 1024 x 768;
* рекомендуемые требования – HDMI с разрешением 1920 x 1080.
* требования к сети:
* пропускная способность более 100 КБ/с (500 КБ/с);
* задержка меннее 100 мс.
* поддержка системой веб-браузера Google Chrome, Microsoft Edge или Mozilla Firefox,

Для корректной работы web-сайта также необходим сервер со следующими характеристиками:

* процессор:
* 8-ядерный x86- или x64-разрядный процессор с тактовой частотой 2,4 ГГц и выше;
* 8-ядерный x64-разрядный процессор с тактовой частотой 3,0 ГГц и выше, поддержка виртуализации;
* память:
* минимальные требования – ОЗУ объемом 8 Гб;
* рекомендуемые требования – ОЗУ объемом 32 Гб и более;
* жесткий диск/Хранилище данных:
* SSD-накопитель объемом не менее 150 Гб для обеспечения быстрой работы приложений и базы данных;
* SSD-накопитель объемом 250 Гб и более.
* требования к сети:
* пропускная способность более 200 мбит/с.

**ЗАКЛЮЧЕНИЕ**

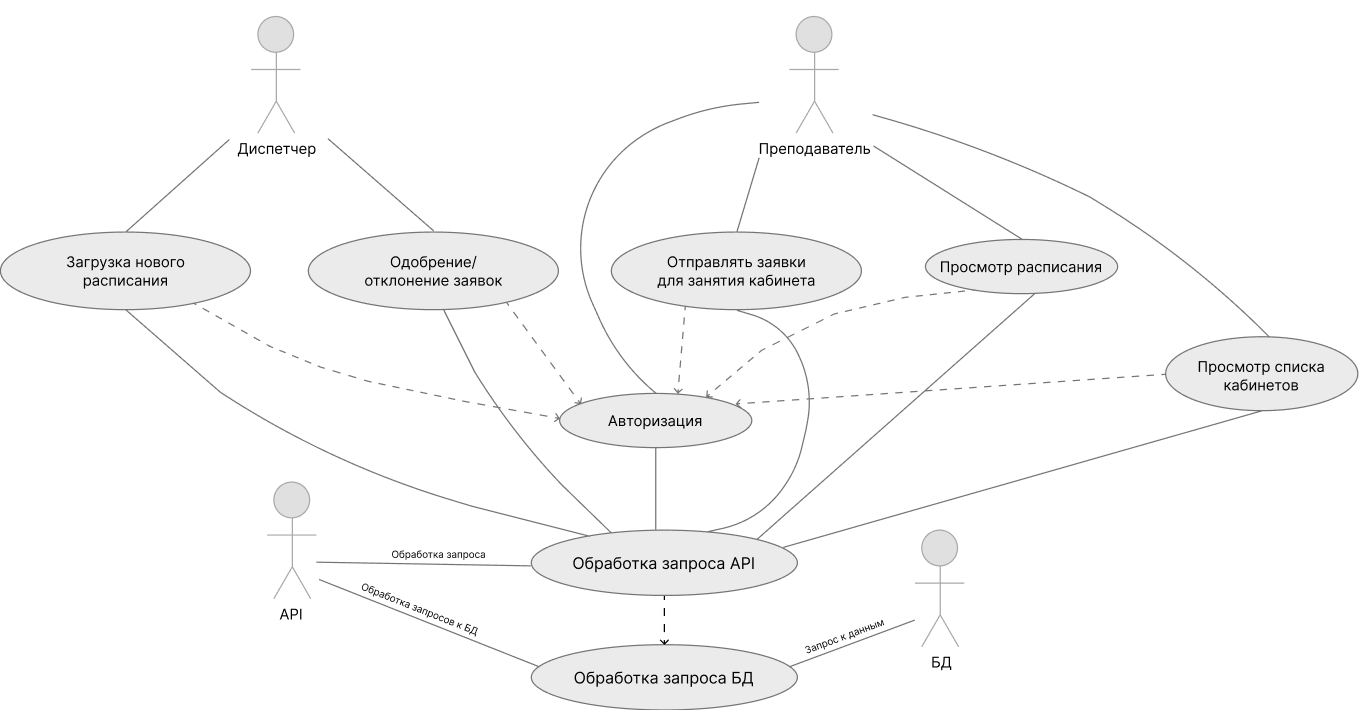
В процессе проектирования приложения учета занятости кабинетов были сформулированы основные принципы и архитектурные решения, что заложило фундамент для дальнейшей разработки приложения. Основная цель проекта заключалась в создании интуитивно понятного web-приложения, которое отвечает потребностям преподавателей и диспетчеров колледжа в управлении кабинетов. С помощью данного приложения пользователи могут легко войти в систему под своим логином и увидеть расписание занятий, а также оставлять заявки на использование определенных кабинетов в заданные даты и пары.

Несмотря на то что проект сейчас находится на стадии проектирования, заложенные принципы и выбранные технологии создают надежную основу для дальнейшей разработки. Важнейшей задачей на следующем этапе станет реализация функций одобрения и отклонения заявок диспетчером, а также тщательно продуманное представление списка кабинетов с их описаниями.

Проектирование системы учета занятости кабинетов дало возможность глубже понять процессы, связанные с разработкой программного обеспечения, а также осознать важность тщательной подготовки и тестирования перед реализацией. В будущем мы сосредоточимся на улучшении пользовательского опыта, обеспечив простоту навигации и эффективность взаимодействия между преподавателями и диспетчерской службой.

**ПРИЛОЖЕНИЕ А**

(Диаграмма вариантов использования**)**



**ПРИЛОЖЕНИЕ Б**

(Логическая модель БД)

