МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Корпоративные информационные системы
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка информационной системы автоматизации управления освещением в кампусе ТГУ»

Студент Е.Д. Федосеев
(И.О. Фамилия) (личная подпись)

Руководитель Кандидат педагогических наук, Доцент, Е.А. Ерофеева
(ученая степень, звание, И.О. Фамилия)

Консультант А. В. Москалюк
(ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Выпускная квалификационная работа посвящена решению проблем с автоматизацией управления освещением в кампусе ТГУ, а также вопросам уменьшения затрат на энергопитание и улучшения процесса сбора статистики по освещению в кампусе и мониторингу действующего освещения.

Структура работы включает в себя введение, три главы, заключение и список литературы.

Введение содержит цели и задачи работы, в нем определена актуальность работы, а также определены объект и предмет исследования.

В первой главе проведен анализ работы АХО ТГУ, разработаны модели «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ», а также рассмотрены и проанализированы аналоги разрабатываемой АИС. Во второй главе были спроектированы логическая модель АИС и логическая модель данных. В третей главе описаны: выбор средств разработки информационной системы, разработка, тестирование, а также подробно описано разработанное web-приложение.

В заключении показаны выводы и результаты о проделанной работе.

Итогом ВКР является web-приложение с мобильной версией сайта, которое позволит, автоматизировано управлять «умным» освещением в кампусе ТГУ, а также собирать по нему статистику.

В работе показано 6 рисунков, 4 таблицы, 7 листингов, список использованной литературы содержит 24 источника. Целый объем ВКР содержит 85 страниц.

Abstract

The subject of the bachelor's thesis is *development of an information system for automation of lighting control on the TSU campus*. It is designed to solve problems with the automation of lighting management on the campus of Togliatti State University, as well as to reduce energy costs and improve the process of collecting statistics on campus lighting and monitoring current lighting.

The aim of the work is to develop an information system that will automate the management of lighting on the TSU campus, as well as allow collecting statistics on data from this information system.

The introduction contains the goals and objectives of the work, it defines the relevance of the work, as well as the object and subject of the study.

In the first chapter, the analysis of the work of the TSU AED is carried out, the "AS IS" and "TO BE" models are developed, as well as the analogues of the information system being developed are considered and analyzed. In the second chapter, the logical model of the information system and the logical data model are designed. The third chapter describes: the choice of information system development tools, development, testing, and also describes in detail the developed web application.

The conclusion presents the conclusions and results of the work done.

The bachelor's thesis has resulted in the developed web-application with a mobile version, which will allow users to automatically manage the "smart" lighting on the TSU campus, as well as collect statistics on it.

The structure of the work consists of an introduction, three chapters, a conclusion, the work contains 6 figures, 4 tables, 7 listings a list of 24 references. The volume of the work is 85 pages.

Оглавление

Введение 4
Глава 1 Функциональное моделирование предметной области 6
1.1 Технико-экономическая характеристика АХО ТГУ 6
1.2 Концептуальное моделирование предметной области 7
1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям 17
1.4 Постановка задачи на разработку проекта создания/внедрения АИС 19
1.5 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ» 21
Глава 2 Логическое проектирование АИС 27
2.1 Выбор технологии логического моделирования АИС 27
2.2 Логическая модель АИС и ее описание 27
2.3 Проектирование базы данных АИС 28
Глава 3 Физическое проектирование АИС 31
3.1 Выбор архитектуры АИС 31
3.2 Выбор технологии разработки программного обеспечения АИС 33
3.3 Выбор СУБД АИС 37
3.4 Разработка физической модели данных АИС 39
3.5 Разработка архитектуры АИС 40
3.6 Разработка серверной части АИС 41
3.7 Разработка клиентской части АИС 47
3.8 Разработка автоматической выгрузки проекта 53
3.9 Тестирование 54
Заключение 56
Список используемой литературы 57
Приложение А Диаграмма прецедентов «КАК ЕСТЬ» 61
Приложение Б Диаграмма прецедентов «КАК ДОЛЖНО БЫТЬ» 62

Приложение В Диаграмма видов деятельности для смены состояния модуля 63
Приложение Г Диаграмма компонентов..... 64
Приложение Д Диаграмма последовательности для процесса смены состояния модуля 65
Приложение Е Логическая модель БД..... 66
Приложение Ж Физическая модель БД 67
Приложение И Диаграмма развертывания 68
Приложение К Структура серверной части..... 69
Приложение Л Пример кода сервиса для выставления активного сценария . 70
Приложение М Пример кода промежуточной функции авторизации 72
Приложение Н Структура клиентской части 73
Приложение П Пример кода функции «ModuleItem» 74
Приложение Р Пример кода карты с главной страницы..... 75
Приложение С Пример кода обработки отправки формы 76
Приложение Т Пример кода для получения пользователя в форму..... 78
Приложение У Пример кода для обработки удаления пользователя 79
Приложение Ф Код файла «.gitlab-ci.yml» 80
Приложение Х Тест-кейсы для кроссбраузерного тестирования 81
Приложение Ц Пример отображения корректной страницы входа..... 82
Приложение Ш Пример отображения графика..... 83
Приложение Щ Пример вывода ошибок для пользователя..... 84
Приложение Э Тест-кейсы для функционального тестирования 85

Введение

Сейчас, в современном мире, невозможно представить развитие какой-либо области без создания автоматизированной информационной системы (АИС), они стали незаменимы во всех областях.

Современные АИС позволяют наладить работу бизнес процессов, автоматизировать рутинную работу, а также помогают со сбором статистики.

Целью выпускной квалификационной работы является разработка информационной системы, которая позволит автоматизировать управление освещением в кампусе ТГУ, а также позволит собираться статистику по данным из этой информационной системы.

Для достижения поставленной цели необходимо составить список задач:

- изучить предметную область;
- провести анализ аналогов;
- составить требования на разработку АИС;
- реализовать АИС;
- протестировать АИС.

Объект исследования – отдел электрического обслуживания АХО тольяттинского государственного университета.

Предметом исследования является автоматизация создания заявок на неработающие электрические приборы, реализация удаленного управления освещением в кампусе ТГУ и автоматизация сбора статистики по осветительным приборам кампуса ТГУ.

Метод исследования:

- анализ;
- моделирование;
- изучение электронных источников.

В выпускной квалификационной работе разбираются вопросы по реализации и разработке информационной системы автоматизации управления освещением в кампусе ТГУ.

В первой главе будет проанализирована деятельность отдела электрического обслуживания ТГУ, составлены требования к разработке информационной системы и проведен сравнительный анализ аналогов.

Во второй главе будут разработаны логические модели для процессов и для данных.

В третей главе будет происходить разработка программного кода информационной системы, анализ возможных технологий реализации и проектирования базы данных. Будут выбраны технологии для разработки информационной системы, а также реализован программный код для ее функционирования.

Глава 1 Функциональное моделирование предметной области

1.1 Технико-экономическая характеристика АХО ТГУ

Подразделение административно-хозяйственного обслуживания тольяттинского государственного университета состоит из нескольких подразделений, которые показаны на рисунке 1.

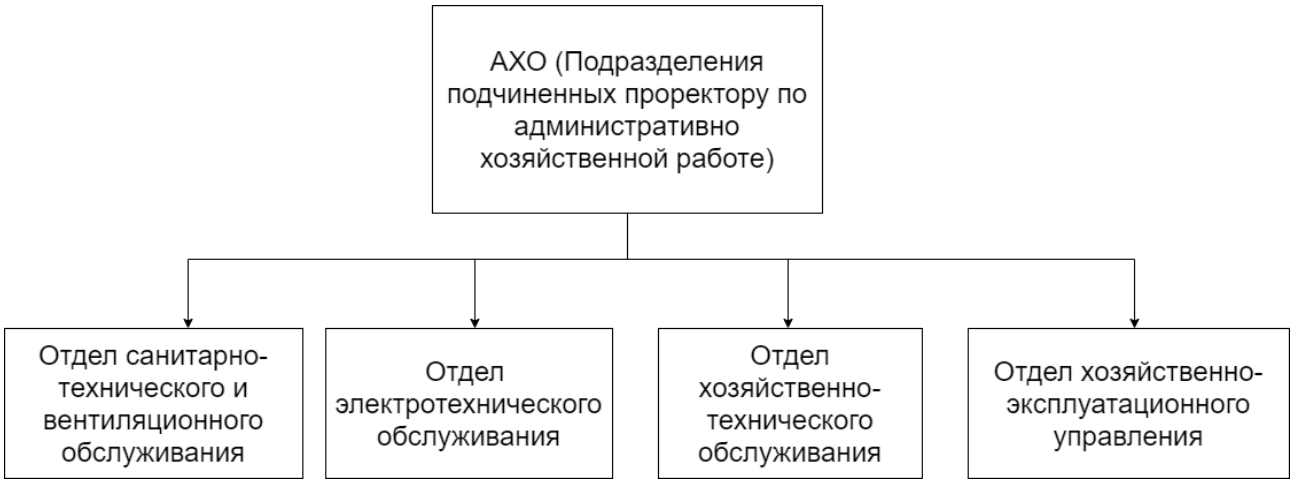

Рисунок 1 – Схема АХО ТГУ

Данные подразделения имеют набор своих обязанностей [7].

Отдел санитарно-технического и вентиляционного обслуживания занимается:

- ремонтом или заменой санитарно-технического оборудования;
- ремонтом или заменой приборов отопления;
- обслуживанием вентиляционного оборудования, шахт вентиляции.

Отдел электротехнического обслуживания занимается:

- ремонтом или заменой ламп и светильников;
- ремонтом или заменой электровыключателей;
- ремонтом или заменой электророзеток.

Отдел хозяйственно-технического обслуживания занимается:

- заменой оконных стекол;
- врезкой замков.

Отдел хозяйственно-эксплуатационного управления занимается:

- ремонтом офисной мебели;
- ремонтом дверей и заменой ее комплектующих;
- ремонтом и установкой крючков, полок, приспособлений для картин, зеркал;
- ремонтом и заменой элементов окон;
- обслуживанием лифтов;
- выдачей материала для утепления окон и дверей;
- контролем работы клининговой компании.

Большинство задач, которые описаны выше начинаются с запроса в службу поддержки АХО ТГУ.

1.2 Концептуальное моделирование предметной области

1.2.1 Выбор технологии концептуального моделирования предметной области

На данный момент времени есть 3 наиболее известные методологии, которые активно используются для концептуального моделирования:

- SADT,
- ARIS eEPC,
- UML [8, 10, 13, 15].

Далее необходимо выделить критерии, по которым будет отбираться наиболее подходящая методология:

- отражение функциональной и поведенческой составляющей предметной области;

– наиболее полное и наглядное описание бизнес-процессов (выразительность модели);
– оптимальное количество типов моделей.

Далее в таблице 1 представлен результат сравнения данных методологий по этим критерия.

Таблица 1 – Сравнение методологий концептуального моделирования

Критерии	SADT	ARIS	UML
Отражение функциональной и поведенческой составляющей предметной области	1	2	3
Наиболее полное и наглядное описание бизнес-процессов (выразительность модели)	1	2	3
Оптимальное количество типов моделей	2	1	3

Сравнение производилось по принципу: для каждого критерия методологии получали от 1 до 3 баллов включительно, причем баллы не могли повторяться, то есть аналог рейтинга.

Таким образом, можно сделать вывод, что UML является лидером среди всех прочих методологий по заданным критериям, потому в ходе дальнейшего концептуального моделирования будет использоваться методология UML. А именно ее следующие диаграммы:

– диаграмма прецедентов,
– диаграмма последовательности,
– диаграмма деятельности.

Также, для концептуального моделирования необходимо выбрать средства реализации, программы, в которых будет происходить разработка UML диаграмм. Для этого необходимо выбрать наиболее популярные

программы и сравнить их. Популярные программы для разработки на данный момент это:

- starUML,
- draw.io,
- Microsoft Visio.

StarUML - программный инструмент моделирования, который поддерживает UML. StarUML ориентирован на UML версии 1.4 и поддерживает одиннадцать различных типов диаграмм. StarUML предоставляет максимальную степень адаптации среды разработки пользователя, предлагая настройку параметров, которые могут влиять на методологию разработки программного обеспечения, проектную платформу и язык.

draw.io - это сервис для создания блок-схем, графиков, диаграмм и других визуальных объектов. Инструмент используется в работе менеджерами, маркетологами, аналитиками, разработчиками и прочими специалистами.

Microsoft Visio - это программное обеспечение для разработки различных диаграмм: блок-схем, организационных диаграмм, планов зданий, поэтажных планов, диаграмм потоков данных, технологических схем, моделирования бизнес-процессов, swim lane блок-схем, 3D-карт.

Чтобы выбрать наиболее подходящий инструмент необходимо написать список требований, который будет выдвинут к ПО, чтобы ПО отвечало данному списку.

Самым важным критерием будет поддержка UML, так как именно эта методология будет использоваться для разработки диаграмм.

Вторым из критериев будет возможность запуска на всех популярных ОС, то есть на linux, windows и macOS.

Третьим важным критерием считается возможность экспорта UML диаграмм в удобный формат, так как наиболее удобным для экспорта

считается формат PDF, то возможность экспорта в PDF и будет вынесено в отдельный критерий.

В современном мире никуда не деться без командной работы, потому современные средства разработки диаграмм должны поддерживать возможность одновременной командной работы.

Большим плюсом будет, если средства разработки будут поддерживать валидацию UML схем, что позволит избежать глупых ошибок.

Также плюсом будет возможность генерации кода на основе UML диаграммы.

Теперь необходимо вынести все критерии в одну таблицу и сопоставить, какие из критериев какими средствами поддерживаются. Это будет представлено в таблице 2.

Таблица 2 – Сравнение средств реализации концептуального моделирования

Критерий	starUML	draw.io	Microsoft Visio
Поддержка UML	+	+	+
Поддержка популярных ОС	+	+	-
Экспорт в PDF	+	+	-
Возможность командной разработки	-	+	-
Валидация UML диаграмм	+	-	-
Генерация кода на основе UML диаграмм	+	-	-
Итого:	5	4	1

Знак «+» означает, что данный критерий поддерживается программой, а знак «-» означает, что программа не поддерживает критерий.

Таким образом, можно заметить, что наибольшему количеству критериев удовлетворяет starUML, к тому же, starUML не выполняет только критерий с командной разработкой, что в рамках ВКР не несет в себе большой надобности.

Таким образом, можно сделать вывод, что программа starUML является наиболее актуальной программой в рамках данной ВКР, также данная программа считается более «профессиональной» на фоне всех остальных.

1.2.2 Моделирование бизнес-процессов предметной области для постановки задачи автоматизированного варианта решения

Отдел электрического обслуживания АХО ТГУ выполняет следующие задачи:

- ремонт или замена ламп и светильников,
- ремонт или замена электровыключателей,
- ремонт или замена электророзеток.

Все эти действия начинаются с непосредственного обращения в службу поддержки АХО или попадают напрямую до данного отдела. Далее задача берется в работу, назначается специалист отдела электротехнического обслуживания, который будет ее выполнять и происходит выполнение задачи.

Разрабатываемая информационная система позволит частично автоматизировать процесс появления задач у отдела электрического обслуживания. А также позволит сократить количество решаемых ими инцидентов, так как если за электрическим оборудованием будет постоянный контроль через разрабатываемую информационную систему, то это увеличит его срок службы, что позволит сэкономить на оборудовании и сократить количество персонала в данном отделе.

Также на данный момент в случае, если кто-то регулярно не выключает свет в аудитории, то об этом порой можно узнать очень не сразу, что приведет к дополнительным расходам, за электричество, а разрабатываемая информационная система позволит избежать таких проблем.

Информацию, которая была получена из анализа предметной области, как правило, используют для создания моделей работы организации. Чаще всего необходимо исследовать модели: «КАК ЕСТЬ» и «КАК ДОЛЖНО

БЫТЬ». Данные модели включают полные информационные и функциональные модели работы организации.

1.2.3 Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Модель «КАК ЕСТЬ» описывает то, как на данный момент в структуре или информационной системе проходят бизнес-процессы.

Основной целью разработки и анализа модели «КАК ЕСТЬ» заключается в том, чтобы определить места в бизнес-процессах, которые замедляют время выполнения задачи или же являются экономически дорогими.

В первую очередь необходимо разработать общую диаграмму деятельности «КАК ЕСТЬ». Данная диаграмма представлена на рисунке 2.

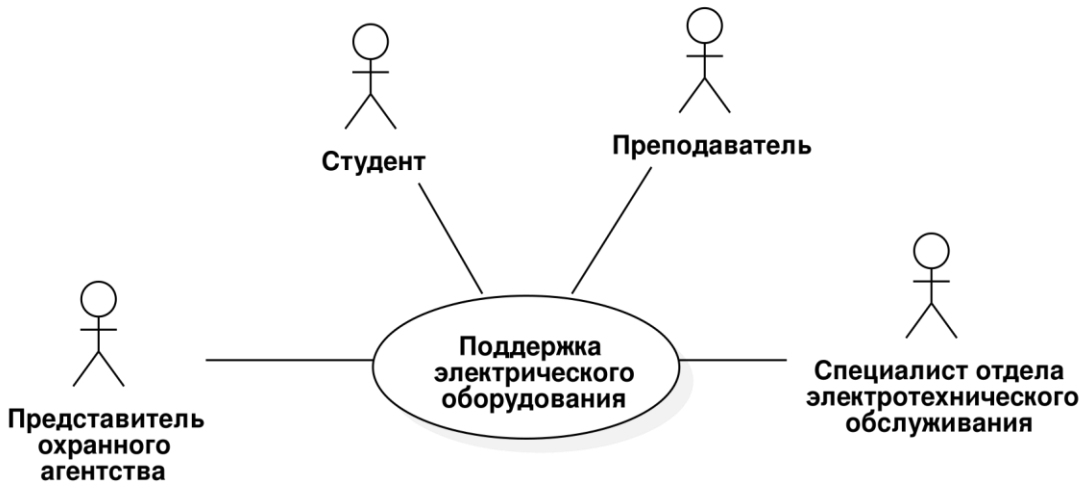

Рисунок 2 – Общая диаграмма «как есть»

На диаграмме представлен один общий прецедент «Поддержка электрического оборудования», в котором принимают участие 4 роли:

- студент,
- преподаватель,
- представитель охранного агентства,
- специалист отдела электрического обслуживания.

Далее необходимо разбить общий прецедент «Поддержка электрического оборудования» на диаграмму прецедентов, как это представлено на рисунке А.1.

Для всех этих прецедентов необходимо описать их основной поток действий и если имеется, то побочный:

а) описание прецедента «Проверка на отключение электроприборов»:
   1) предусловие: дождаться, когда рабочий день закончится у большего числа людей;
   2) основной поток:
      – пройти по всем этажам;
      – проверить, что свет везде выключен;
      – проверить, что никакой электроприбор сейчас не включен;

б) описание прецедента «Ремонт ламп и светильников»:
   1) предусловия: получить заявку от СП АХО;
   2) основной поток:
      – получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
      – получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
      – убедиться, что лампа или светильник подлежит ремонту;
      – провести ремонт лампы или светильника;

в) описание прецедента «Замена ламп и светильников»:
   1) предусловия: получить заявку от СП АХО;
   2) основной поток:
      – получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
      – получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
      – убедиться, что лампа или светильник не подлежит ремонту;

- провести замену лампы или светильника;

г) описание прецедента «Ремонт электровыключателей»:

1) предусловия: получить заявку от СП АХО;
2) основной поток:
   - получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
   - получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
   - убедиться, что электровыключатель подлежит ремонту;
   - провести ремонт электровыключателей;

д) описание прецедента «Замена электровыключателей»:

1) предусловия: получить заявку от СП АХО;
2) основной поток:
   - получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
   - получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
   - убедиться, что электровыключатель не подлежит ремонту;
   - провести замену электровыключателей;

е) описание прецедента «Ремонт электророзеток»:

1) предусловия: получить заявку от СП АХО;
2) основной поток:
   - получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
   - получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
   - провести ремонт электророзеток;

ж) описание прецедента «Замена электророзеток»:

1) предусловия: получить заявку от СП АХО;

2) основной поток:
   - получить полную информацию об инциденте (корпус, кабинет, коридор и тд.);
   - получить доступ к помещению, если необходимо, это делается через студента/сотрудника оставившего заявку;
   - убедиться, что электророзетка не подлежит ремонту;
   - провести замену электророзеток;

з) описание прецедента «Написание в поддержку о нерабочем электроприборе»:
   1) предусловие: найти инцидент, с которым необходимо обратиться в СП АХО;
   2) основной поток:
      - перейти на сайт СП АХО;
      - оставить заявку о произошедшем инциденте;
      - дождаться закрытия заявки;
      - закрыть заявку после проверки работы на выполнение;
   3) альтернативный поток:
      - перейти на сайт СП АХО;
      - оставить заявку о произошедшем инциденте;
      - получить звонок от работника СП АХО с уточнением местоположения происшествия или его деталях;
      - дождаться закрытия заявки;
      - закрыть заявку после проверки работы на выполнение.

Из описанного выше списка прецедентов можно сделать вывод, что для большинства этих прецедентов предусловием является заявка из СП АХО. Это приводит к тому, что специалист отдела электрического обслуживания не сразу получает информацию об инциденте, также не всегда информация доходит, так как не все знают, что необходимо писать в СП АХО при данном инциденте. Помимо того, что это происходит с задержкой, в самом решении

инцидента принимает участие сотрудник или студент, что тоже является не логичным [2].

1.2.4 Обоснование необходимости автоматизированного варианта решения и формирование требований к новой технологии

На основе разработанных выше моделей бизнес-процессов «КАК ЕСТЬ» можно сделать вывод, что некоторые из них подлежат оптимизации или автоматизации.

Далее необходимо разработать список требований, которые должны быть выдвинуты к информационной системе. Требования к информационной системе будут строиться на основе системы FURPS+:

а) functionality, функциональность:
   1) отчет по времени, в каком состоянии (вкл/выкл) провели все приборы за месяц;
   2) возможность через UI интерфейс изменять состояние ламп или светильников в аудитории;
   3) возможность администрирования ламп или светильников через UI интерфейс;
   4) возможность администрирования пользователей через UI интерфейс;
   5) возможность скачивания логов сервера через UI интерфейс;
   6) возможность увидеть в UI интерфейсе нерабочие лампы или светильники;

б) usability, удобство использования:
   1) интуитивно понятный UI интерфейс;
   2) схематичное отображение выбранного этажа в виде карты;
   3) поиск аудиторий в интерфейсе;
   4) версия для мобильного телефона, с урезанным функционалом;

в) reliability, надёжность:
   1) разные права доступа у пользователей с разными ролями;
   2) валидация вводимой информации;

г) performance, производительность:

1) обработка сортировок и выборок приоритетно на сервере, а не на клиентской части приложения;
2) работа с лампами или светильниками только на сервере, а не на клиенте;

д) supportability, поддерживаемость:

1) возможность масштабирования;
2) наличие документации с первичной настройкой, для простого первичного запуска;
3) документация всех API точек для работы с данным приложением.

Таким образом, были сформированы требования к разрабатываемой информационной системе [11].

1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям

1.3.1 Определение критериев анализа

Исходя из критериев, описанных выше в системе FURPS+ можно составить список основных критериев анализа, которые должны выполняться разрабатываемой информационной системой.

Приложение должно иметь график отчетов по состояниям ламп или светильников, для того, чтобы вести учет статистики, не происходит ли злоупотребление электрическими приборами.

Должна быть мобильная и компьютерная версия web-приложения. Так как с телефона удобно включать/выключать электроприборы, но на компьютерной версии будет больше функциональных возможностей.

Модуль, который не отвечает должен помечаться в системе, чтобы незамедлительно починить его.

В web-приложении, **в компьютерной версии должна присутствовать схема этажа**, чтобы конечный пользователь мог без проблем ориентироваться в том, где не работает или где вышел из строя светильник.

У разных пользователей должны быть разные права, так например, у специалиста отдела электрического оборудования не должно быть доступа для редактирования информации о пользователях, а также ему не нужны отчеты.

Так как ламп или светильников на каждом из этажей много, то **должна быть строка поиска по** лампам или светильникам.

Приборы должны объединяться в этажи, а этажи в здания, чтобы облегчить поиск конкретного устройства на этаже.

Приложение должно иметь открытое и публичное API, чтобы система могла интегрироваться с другими сервисами ТГУ в будущем.

1.3.2 Сравнительная характеристика существующих разработок

Далее необходимо разобрать, удовлетворяют ли существующие решения на рынке вышеописанные критерии.

Для этого необходимо составить список популярных готовых решений:

– Умный дом Яндекса,
– IOS Умный дом,
– Mi home.

Далее необходимо определить, какие из критериев выполняются в данных программных решениях. Результаты проверки можно увидеть в таблице 3.

Таблица 3 – Результаты сравнения аналогов

Требование/Аналог	Умный дом Яндекса	IOS Умный дом	Mi home
Приложение должно иметь график отчетов по модулям	0	0	0
Мобильная и компьютерная версия	0.5	0.5	0.5

Требование/Аналог	Умный дом Яндекса	IOS Умный дом	Mi home
web-приложения			
Модуль, который не отвечает должен помечаться в системе	0	0	0
В компьютерной версии должна присутствовать схема этажа	0	0	0
У разных пользователей должны быть разные права	0	0	0.5
Строка поиска по электроприборам	0	0	0
Приборы должны объединяться в этажи, а этажи в здания	0.5	0.5	0.5
Приложение должно иметь открытое и публичное API	1	1	1
Итого:	2	2	2.5

Таблица заполнена значениями 1, 0.5 и 0. Данные числовые значения обозначают:

- 1 – удовлетворяет условие;
- 0.5 – удовлетворяет условие не полностью;
- 0 – не удовлетворяет условие.

Исходя из того, на сколько, соответствуют требованиям данные системы можно сделать вывод, что они прекрасно выполняют все требования «умного дома», но, к сожалению, не подходят для управления освещением в кампусе ТГУ, так как сильно не соответствуют выдвинутым требованиям.

1.4 Постановка задачи на разработку проекта создания/внедрения АИС

Исходя из недостатков аналогов, описанных выше можно сделать вывод, что информационная система должна либо иметь собственные «умные осветительные приборы», либо контактировать с уже

разработанными приборами. Для такого подхода во всей информационной системе должен быть модуль по работе с «умными осветительными приборами», что позволит без проблем садаптировать ее к любым таким приборам, будь это «умная лампочка» Яндекс или самостоятельно разработанная лампочка от сторонних разработчиков. Такой подход позволит создать собственную информационную систему, как на базе уже готовой, так и с самого нуля.

Разрабатываемая информационная система должна решать следующие проблемы:

- отсутствие информации об осветительных приборах в данный момент времени;
- отсутствие информации о работе осветительных приборов;
- задержка информирования, о нерабочем осветительном приборе;
- включенный свет ночью и трата электричества.

Также разрабатываемая информационная система должна иметь мобильную и компьютерную версии, в которых была бы возможность управления осветительными приборами.

Разрабатываемая информационная система должна иметь подробную API документацию, чтобы в случае масштабирования никакая информация о разработках системы не была утеряна или переиначена.

Информационная система должна иметь иерархическую структуру данных:

- здание,
- этаж,
- осветительный прибор.

Такая структура позволит без проблем определить, где находится тот или иной прибор. Также каждый осветительный прибор должен схематически отображать кабинет, то есть должен содержать в себе информацию о местоположении и размерах кабинета.

Разрабатываемая информационная система является достаточно гибкой, за счет того, что будет иметь отдельный модуль для работы с осветительными приборами, таким образом, внедрение данной информационной системы будет достаточно простым, так как она строго не привязана к реализации «умного» освещения и может работать с чем угодно.

1.5 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

Далее необходимо разработать модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ». Для этого необходимо аналогично составить общую use case диаграмму, представленную на рисунке 3.

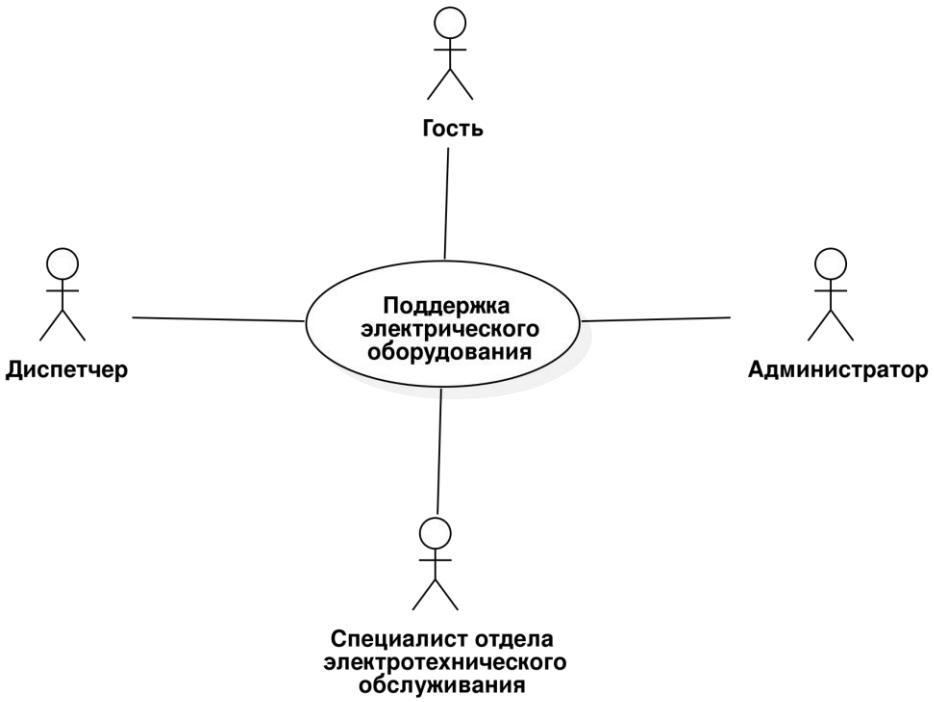

Рисунок 3 – Общая диаграмма «как должно быть»

На этой диаграмме уже есть 4 роли:

– гость – роль пользователя, который ничего не может делать в системе, кроме как просмотра списка модулей и их состояний;

– диспетчер – роль, которая позволяет изменять состояния модулей и просматривать отчеты;
– специалист отдела электрического оборудования – роль пользователя для просмотра состояния модулей и изменения состояний;
– администратор – роль, которая принадлежит администратору сервиса, где будет доступ к редактированию информации приложения, а также доступ к взаимодействию с пользователем.

Далее необходимо расписать подробно этот прецедент, как показано на рисунке Б.1.

Как видно на схеме, то прецедентов стало больше, но это не значит, что не произошло оптимизации, так как добавились новые возможности, которых ранее не было.

Далее необходимо для изменившихся прецедентов написать основной поток действий, а также побочный, если имеется:

а) описание прецедента «Редактирование информации о пользователях»:
   1) предусловие: быть аутентифицированным под ролью «Администратор»;
   2) основной поток данных:
      – перейти на вкладку «Администрирование»;
      – открыть ссылку «Пользователи»;
      – произвести редактирование пользователей;
      – нажать кнопку «Сохранить изменения»;

б) описание прецедента «Просмотр логов»:
   1) предусловие: быть аутентифицированным под ролью «Администратор»;
   2) основной поток данных:
      – открыть вкладку «Администрирование»;

- открыть ссылку «Логи сервера»;
- нажать на файл логов для скачивания;

в) описание прецедента «Создание осветительных приборов»:

1) предусловие: быть аутентифицированным под ролью «Администратор»;
2) основной поток данных:
   - открыть вкладку «Администрирование»;
   - открыть ссылку «Изменение данных о модулях»;
   - выбрать здание и этаж;
   - в последней строке написать данные нового модуля;
   - нажать кнопку «Сохранить изменения»;

г) описание прецедента «Создание пользователей»:

1) предусловие: быть аутентифицированным под ролью «Администратор»;
2) основной поток данных:
   - открыть вкладку «Администрирование»;
   - открыть ссылку «Пользователи»;
   - в последней строке написать данные нового пользователя;
   - нажать кнопку «Сохранить изменения»;

д) описание прецедента «Редактирование информации об осветительных приборах»:

1) предусловие: быть аутентифицированным под ролью «Администратор»;
2) основной поток данных:
   - открыть вкладку «Администрирование»;
   - открыть ссылку «Изменение данных о модулях»;
   - выбрать здание и этаж;
   - изменить данные модуля;
   - нажать кнопку «Сохранить изменения»;

е) описание прецедента «Получение списка нерабочих модулей»:

1) предусловие: быть аутентифицированным под ролью «Специалист отдела электротехнического обслуживания» или «Администратор»;
2) основной поток данных:
   - открыть вкладку «Диспетчер»;
   - выбрать корпус и этаж;
   - найти нерабочие модули, которые помечены в системе.

ж) описание прецедента «Ремонт ламп и светильников»:

1) предусловие: быть аутентифицированным под ролью «Специалист отдела электротехнического обслуживания»;
2) основной поток данных:
   - в системе увидеть модуль, который помечен, как не отвечающий;
   - пойти ремонтировать лампу или светильник;

з) описание прецедента «Замена ламп и светильников»:

1) предусловие: быть аутентифицированным под ролью «Специалист отдела электротехнического обслуживания»;
2) основной поток данных:
   - в системе увидеть модуль, который помечен, как не отвечающий;
   - пойти ремонтировать лампу или светильник;

и) описание прецедента «Редактирование состояний осветительных приборов»:

1) предусловие: быть аутентифицированным под ролью «Диспетчер» или «Администратор»;
2) основной поток данных:
   - открыть вкладку «Диспетчер»;
   - в списке слева найти модуль;

– выбрать состояние;
– применить состояние на модуль;

3) альтернативный поток:
– открыть вкладку «Диспетчер»;
– в поиске вбить название модуля;
– выбрать состояние;
– применить состояние на модуль;

4) альтернативный поток:
– открыть вкладку «Диспетчер»;
– на схеме найти модуль;
– выбрать состояние;
– нажать на модуль двойным кликом;

к) описание прецедента «Просмотр состояний осветительных приборов»:

1) предусловие: быть аутентифицированным;

2) основной поток данных:
– открыть вкладку «Диспетчер»;
– выбрать здание и этаж;

л) описание прецедента «Просмотр отчетом о режимах»:

1) предусловие: быть аутентифицированным под ролью «Диспетчер» или «Администратор»;

2) основной поток данных:
– открыть вкладку «Аналитика».

Прецеденты «Ремонт электроразеток» и «Замена электроразеток» не претерпели изменений.

Исходя из построенной диаграммы прецедентов, можно сделать вывод, что необходимо составить диаграмму видов деятельности для процесса смены состояния модуля, чтобы наглядно увидеть, как должна работать

информационная система. Диаграмма для данного процесса представлена на рисунке В.1.

Исходя из анализа данного процесса, можно увидеть, что в случае использования разрабатываемой АИС все действия с модулями проходят на уровне UI интерфейса, а система сама отправляет запросы модулям без дополнительных требований от пользователей.

Выводы по главе 1

Анализ предметной области и бизнес-процессов показал, что работа с электрическими приборами проходит крайне рутинно, а также имеет задержку при попадании информации от АХО до специалиста отдела электрического обслуживания.

Также на основе анализа аналогов было выявлено, что среди существующих аналогов отсутствуют решения, которые в полной мере будут отвечать требованиям, выдвинуты к АИС. Также аналоги являются достаточно дорогими, что делает их использование менее выгодным.

Помимо этого были разработаны начальные требования к разрабатываемой информационной системе.

Глава 2 Логическое проектирование АИС

2.1 Выбор технологии логического моделирования АИС

Логическое моделирование предметной области используется для описания процесса и его развития, создаются сценарии, которые направлены на увязку логической последовательности происходящих событий. Логическое моделирование представляет собой анализ логики развития прогнозируемого объекта.

Логическое моделирование подразумевает создание логической модели информационной системы и модели ее базы данных на основе концептуальной модели «КАК ДОЛЖНО БЫТЬ», описанной в первой главе.

Для того, чтобы разработать логическую модель АИС необходимо более подробно разобрать процесс использование АИС. Для этой цели необходимо составить диаграмму последовательности, на основе которой будет сделан вывод о том, как проходит работа АИС и как происходит взаимодействие внутри АИС.

В качестве CASE-средств будет использоваться StarUML.

2.2 Логическая модель АИС и ее описание

В первую очередь необходимо разработать диаграмму компонентов АИС, для того, чтобы можно было увидеть, то, как должны взаимодействовать между собой разные компоненты АИС, данная диаграмма представлена на рисунке Г.1.

Исходя из этой диаграммы, можно увидеть, как происходит взаимодействие между разными частями разрабатываемой информационной системы. Так, например, можно увидеть, что приложение работает сразу и с базой данных и с компонентом для работы с модулями осветительных приборов. Данный подход, по реализации компонента для работы с

модулями отдельно позволит без проблем использовать абсолютно любое умное освещение, которое будет удовлетворять своим функционалом конечных пользователей. Это позволит использовать готовое умное освещение или разработать собственное, в зависимости от того, что будет более выгодно в плане реализации [3, 12].

Далее необходимо разработать диаграмму последовательности для процесса смены состояния модуля, но теперь в этой диаграмме будет отображена и логика компонента для работы с модулями. Данная диаграмма изображена на рисунке Д.1.

Исходя из данной диаграммы наглядно видно, что вся информационная система работает с модулями через компонент для работы с модулями, что позволяет в случае смены реализации модулей просто исправить код только данного компонента. Также такой подход позволит в дальнейшем использовать разные осветительные приборы, от разных разработчиков, которые имеют разную реализацию. Так как есть компонент для работы с модулями, который является некой прослойкой между существующим API модуля и интерфейсом необходимым для разрабатываемой информационной системы.

2.3 Проектирование базы данных АИС

2.3.1 Выбор технологии проектирования БД АИС

Далее необходимо разобрать логическую модель данных, которая будет отображаться данные и отношения между ними, на основе этой модели будет строиться дальнейшая разработка БД.

Необходимо выбрать средства для разработки логической модели АИС. Для этого существуют два наиболее популярных варианта UML и IDEF1X.

UML на данный момент является популярной технологией для моделирования баз данных. Модели на основе UML являются отличным визуальным инструментом для проведения мозгового штурма, когда еще нет

четкой структурированной задачи и необходимо увидеть изъяны БД будущей информационной системы.

IDEF1X является методологией для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Основным преимуществом IDEF1X перед своим конкурентом UML является то, что он имеет строгую спецификацию, которая описывает абсолютно все кейсы [5, 6].

Таким образом, можно сделать вывод, что для построения логической модели данных необходимо использовать IDEF1X.

2.3.2 Разработка логической модели данных АИС

Необходимо разработать логическую модель БД в соответствии со спецификацией IDEF1X. Данная модель изображена на рисунке Е.1.

В данной модели можно увидеть, что существует общая структура «Модель», там описаны поля, которые будут находиться во всех будущих моделях (таблицах): «Дата создания», «Дата редактирования».

Также были созданы связи между осветительными приборами и состояниями (сценариями) данных приборов.

Были настроены связи для приборов, теперь каждый прибор привязан к этажу, а этаж к корпусу. Данная архитектура позволит в дальнейшем осуществлять простой поиск по БД, если будет необходимо найти все модули конкретного этажа.

Для таблицы «Осветительные приборы» было добавлено поле «Ошибка», на основе этого поля в дальнейшем можно будет понять, отвечает ли модуль на запросы или нет.

Таблица «Сценарии» содержит состояния модулей, название «Сценарии» было выбрано, так как в дальнейшем система может не просто сменять состояние освещения, а, например, сменять цвет или яркость, потому название «сценарий» является более предпочтительным.

Также таблица «Роли» была создана, чтобы в случае масштабирования можно было без правок в коде изменять название существующих ролей или добавлять новые.

Выводы по главе 2

В ходе данной главы было разработано несколько диаграмм, которые позволили понять принципы работы разрабатываемой АИС, а также понять, что разрабатываемая АИС может работать даже с разными осветительными приборами, например, от разных производителей.

Также в ходе данной главы была разработана логическая модель БД, которая полностью удовлетворяет всем требованиям разрабатываемой информационной системы.

Глава 3 Физическое проектирование АИС

3.1 Выбор архитектуры АИС

На данном этапе необходимо выбрать архитектуру для разрабатываемой информационной системы. Сейчас существует 3 наиболее популярные архитектуры:

- файл-сервер,
- клиент-сервер,
- многоуровневая система.

Файл-сервер – это архитектурное решение, которое использует вычислительную мощность клиентского компьютера, а сервер в этой архитектуре только возвращает данные из файлов. Этот подход хорош тем, что при увеличении числа пользователей растет вычислительная мощность информационной системы. Однако такой подход не очень хорош. Из-за того, что вычисления происходят на стороне клиента, то передача большого объема данных может привести к тому, что сеть будет перегружена.

Клиент-сервер – это архитектурное решение, которое предназначено для решения проблем файл-сервер архитектуры. В случае с файл-сервер архитектурой считается обычной практикой использовать выделенные сервера баз данных и выполнять в них запросы и SQL функции и агрегации. В то время, как клиент-сервер использует двухуровневую модель, в которой клиент обращается к другим заранее реализованным методам сервера и получает ответ. При таком подходе предполагается, что интерфейс не взаимодействует с БД напрямую, а через сервер, где расположены компоненты управления данными.

Многоуровневая архитектура – это продолжение архитектуры клиент-сервер, в котором добавляются новые уровни. Самый распространенный вариант – это три уровня:

- нижний уровень представляет собой приложение клиентов, выделенные для выполнения функций и логики представлений и имеющие программный интерфейс для вызова приложения на среднем уровне;
- средний уровень представляет собой сервер приложений, на котором выполняется прикладная логика и с которого логика обработки данных вызывает операции с базой данных;
- верхний уровень представляет собой удаленный специализированный сервер базы данных, выделенный для услуг обработки данных и файловых операций (без использования хранимых процедур).

К сожалению, на данный момент в России многоуровневая архитектура еще не так сильно развита и самую большую популярность имеет архитектура клиент-сервер.

Таким образом, из 3 рассмотренных вариантов и сформулированных выше требований можно сделать вывод, что архитектура файл-сервер является не достаточно надежной и масштабируемой, а также ее использование в разрабатываемой информационной системе приведет к тому, что она будет сильно нагружать сеть.

В использовании многоуровневой архитектуры, в рамках разрабатываемой информационной системы, нет необходимости, так как разрабатываемая информационная система подразумевает использование только в рамках ТГУ, что означает, что база данных не будет содержать большое количество записей, а значит выносить ее на отдельный сервер нет необходимости.

Таким образом, можно сделать вывод, что наиболее подходящей архитектурой для разрабатываемой информационной системы является архитектура «клиент-сервер». К тому же данная архитектура доминирует на российском рынке, а значит, ее дальнейшее поддержание будет обходиться

дешевле, так как найти специалиста с нужными навыками будет в разы проще.

3.2 Выбор технологии разработки программного обеспечения АИС

Исходя из архитектуры (клиент-сервер) разрабатываемой информационной системы, можно сделать вывод, что для ее разработки потребуются технологии для разработки серверной логики и разработки клиентской логики.

Для написания клиентской части существует 2 наиболее распространенных подхода:

– шаблонизатор,

– SPA.

Шаблонизатор – это подход к разработке, когда на основе существующих html шаблонов создаются html странички. Шаблонизаторы создают html непосредственно на сервере на основе данных из БД.

SPA – это single page application, оно использует 1 html страницу, как шаблон, а на основе JavaScript кода создает целые html страницы и динамически изменяет контент на всем сайте.

Далее необходимо сопоставить данные технологии и оценить, какая из них больше подходит для разрабатываемой информационной системы. Сравнение данных технологий изображено на таблице 4, где 0 – это «несоответствие критерию», а 1 – это «полное соответствие критерию».

Таблица 4 – Сравнение подходов реализации клиентской части web-приложения

Критерий/Подход	Шаблонизатор	SPA
Возможность SSR	1	1
Масштабируемость	0.5	1

Продолжение таблицы 4

Критерий/Подход	Шаблонизатор	SPA
Отсутствие перезагрузки страниц	0.5	1
Скорость работы	0.5	1
SEO оптимизация	1	1 (При подключении SSR)
Отсутствие зависимости от сервера	0	1
Итого	3.5	6

Таким образом, на основе сравнения можно сделать вывод, что SPA приложения являются более гибкими и масштабируемыми, чем шаблонизаторы. Также отсутствие зависимости клиентской части от технологий серверной части позволит писать независимый код и если потребуется переписать серверную часть, то необходимости переписывать клиентскую не будет, если новая серверная часть будет исполнять тот же API, что и прошлая.

Также важно отметить, что SPA позволяют легко динамически сменять состояния в приложении, что является очень удобным при разработке, а большое количество готовых библиотек позволит не писать весь код с нуля, а использовать уже готовые решения.

Среди технологий, которые реализуют подход SPA, на данный момент существует 3 наиболее популярных:

- React,
- Angular,
- Vue.

React – это JavaScript библиотека разработанная компанией Facebook в 2013 году.

У React есть список преимуществ:

- JSX синтаксис, данный синтаксис очень похож на HTML синтаксис за некоторыми исключениями, что позволит быстро его освоить;
- имеет поддержку SSR;
- за счет утилиты CRA имеет отличную и удобную поддержку PWA;

– приложение может быть реализовано как на JavaScript, так и на TypeScript;
– простой переход между версиями, как правило, Facebook предоставляет средства для автоматического перехода к новой версии;
– помимо web-разработки данную библиотеку можно использовать и для мобильной разработки, ReactNattive;
– есть несколько вариантов написания стилей (Например: css и css-in-js);
– наличие документации на русском языке [18].

Angular – это фреймворк, разработанный командой из google, а также разработчиками из других компаний.

У Angular есть свои преимущества:

– CLI утилита для автоматического создания компонентов;
– заранее продуманная архитектура для большой масштабируемости проекта;
– MVVM (Model-View-ViewModel), которая позволяет разработчикам раздельно работать в одном разделе приложения с использованием одного и того же набора данных [16].

Vue – это JavaScript фреймворк, созданый в 2013 году.

У Vue есть свои преимущества:

– обширная документация на русском языке;
– простой синтаксис, который похож на стандартный HTML, из-за чего данный фреймворк имеет низкий порог входа;
– маленький размер конечного веса файла [19].

Исходя из описанных преимуществ, можно сделать вывод, что использование React является самым перспективным вариантом. Так как большой разницы между данными фреймворками нет. Использование в проекте React имеет большой потенциал на создание мобильного

приложения. Помимо этого использование данной библиотеки не накладывает ограничений на использование JavaScript или TypeScript, а также предоставляет выбор в написании стилей.

Далее необходимо выбрать технологию для реализации серверной части, на данный момент существует большое количество технологий для разработки серверной логики. Далее будут приведены примеры наиболее популярных, на данный момент:

- JavaEE,
- Django,
- express,
- ASP.NET,
- Ruby on Rails,
- Spring [17].

Для разрабатываемой информационной системы необходима достаточно легковесная технология, так как разрабатываемая информационная система не планируется, как система управлением освещением более, чем всеми кампусами ТГУ нет необходимости в том, чтобы создавать тяжеловесные сервера. Потому технология spring не подойдет, так как приложения на данной технологии получаются очень тяжелыми.

Далее хочется отметить, что разрабатываемая информационная система должна быть легко поддерживаемой, а значит и популярной среди большинства профессиональных разработчиков. Однако опрос от stackoverflow за 2020 год показал, что технологии Django и Ruby on Rails пока что являются не самыми популярными среди разработчиков, потому они тоже не могут быть использованы для разрабатываемой информационной системы [23].

Таким образом, остались технологии:

- JavaEE,

– Express,
– ASP.NET.

Эти три технологии используют языки: Java, JavaScript, C#, соответственно списку выше [14].

Также хочется заметить, что языки JavaScript и C# поддерживают асинхронные операции. Это позволяет очень сильно оптимизировать обработку запросов. Так как данная технология позволяет не останавливать основной поток выполнения, например, при запросе в базу данных нет необходимости ждать ответа и тем самым останавливать весь поток, есть возможность выполнять другие операции, пока ответ от БД не придет. Потому данные языки будут более предпочтительны, нежели Java, так как разрабатываемая система будет делать достаточно много запросов с сервера в БД и на осветительные приборы.

При выборе между express и ASP.NET можно также заметить, что node.js имеет очень большой инструментарий для разработчиков. Данный плюс позволяет не тратить большое число времени на решение крайне тривиальных задач, а сосредоточиться на создании функционала именно информационной системы. Также важно отметить, что написание серверной части на JavaScript позволяет реализовать систему в разы быстрее, так как код, написанный для одной части приложения, можно будет использовать как на клиентской части, так и на серверной [21].

3.3 Выбор СУБД АИС

Далее необходимо выбрать СУБД для разрабатываемой информационной системы. На данный момент есть 2 популярных вида СУБД: SQL и NoSQL.

В SQL (Реляционная) БД данные хранятся в таблице. Данные строго структурированы и связаны друг с другом. В таблицах есть строки и

столбцы. Строки – это записи в БД. Столбцы – поле с назначенным типом данных. В каждой ячейке информация записана по шаблону.

В NoSQL (Нереляционная) БД данные хранятся в коллекции. Данные не имеют четких связей и структуры. Вместо этого данные БД хранят документы, которые могут представлять собой любую структуру. В отличие от реляционных БД не поддерживает SQL.

Нереляционные БД используются в проектах, где структура заранее не ясна или в быстро растущих информационных системах, например: в играх, CMS системах и так далее.

Реляционные БД используются, когда есть заранее спроектированная структура данных с готовыми схемами.

Таким образом, можно сделать вывод, что для разрабатываемой информационной системы наиболее подходящим вариантом будет реляционная БД [1].

Среди реляционных СУБД есть несколько наиболее популярных, согласно опросу от stackowerflow за 2020 год, для анализа будет выбрано 3:

- MySQL,
- PostgreSQL,
- Microsoft SQL server.

Далее необходимо сравнить данные СУБД.

СУБД Microsoft SQL server – является достаточно дорогой платформой. Также данная СУБД устанавливается только на операционную систему Windows, что накладывает дополнительные, необоснованные требования на сервер, из-за чего данная СУБД не подходит.

MySQL – это свободная реляционная СУБД. Разработку и поддержку MySQL осуществляет корпорация Oracle. Достаточно быстрая СУБД, которая имеет простой SQL. Однако MySQL поддерживает далеко не все возможности стандарта SQL. Разработчики данной СУБД решили, что в угоду удобства можно пренебречь некоторыми стандартами SQL [9].

PostgreSQL – это свободная объектно-реляционная СУБД с открытым исходным кодом. Данная СУБД поддерживает стандарты SQL и старается реализовать все его новые стандарты. PostgreSQL поддерживает свои, создаваемые форматы данных и работу с JSON. Имеет достаточно большой размер для своих таблиц (32TB) и неограниченный размер для базы данных [4].

Таким образом, на основе описанных выше СУБД можно сделать вывод, что PostgreSQL является наиболее подходящим вариантом для использования в разрабатываемой информационной системе.

3.4 Разработка физической модели данных АИС

Далее необходимо разработать физическую модель данных, которая будет использоваться в разрабатываемой информационной системе. Данная модель изображена на рисунке Ж.1.

В данной модели учтены все требования к разрабатываемой информационной системе. Можно сделать вывод, что модели должны иметь каскадное удаление, а также каскадное обновление, в случае, если родители были удалены или изменены.

Также была создана таблица для сопоставления, для того, чтобы удалить связь многие ко многим, данная таблица «ModuleScenarios» позволяет собирать статистику за промежуток времени по сценариям.

Также была создана таблица с ролями, чтобы не добавлять их в программный код.

Данная модель позволяет в дальнейшем развивать разрабатываемую информационную систему до нескольких корпусов за счет таблицы «Buildings».

Таблица «Modules» будет содержать в себе все осветительные приборы, так как в рамках разрабатываемой информационной системы есть возможность использовать ее реализацию не только для «умного»

освещения. Были добавлены новые столбцы. Столбцы «x» и «y» отвечают за положение аудитории на карте. Столбцы «width» и «height» отвечают за размеры аудитории на схеме карты. За счет столбца «ip» можно всегда определить, куда необходимо отправлять запрос на смену состояния (сценария).

Также все модели имеют столбцы «updatedAt» и «createdAt», за счет которых можно без проблем определить дату создания или последнего редактирования у записи.

3.5 Разработка архитектуры АИС

Для разработки архитектуры необходимо построить общую диаграмму развертывания АИС. Данная диаграмма представлена на рисунке И.1.

На основе данной диаграммы развертывания можно увидеть, что обмен информацией между сервером и модулем будет происходить за счет http протокола. А клиентская и серверная часть будут взаимодействовать друг с другом по средствам http/https протокола и 80 или 443 порта. Каждый модуль должен будет обязательно иметь какой-либо загрузочный файл, за счет которого будет происходить взаимодействие с ним через http протокол.

Также можно обратить внимание, что сервер и БД находятся на одной физической машине, что позволяет легко транспортировать разрабатываемую АИС с одной машины на другую.

Помимо этого сервер запускается внутри Node.js, что позволяет запустить его на абсолютно любой ОС в будущем, будь то Linux, Windows или MacOS.

3.6 Разработка серверной части АИС

3.6.1 Разработка структуры серверной части

В первую очередь необходимо разработать структуру для сервера. Она показана на рисунке К.1.

Папка «config» хранит в себе конфигурационные файлы сервера, для создания конфигурации используется библиотека «nconf», которая позволяет получать переменные, как из env, так и из конфигурационного json файла. Также данная папка будет содержать коннект к базе данных. Содержимое главного файла конфигурации изображено в листинге 1.

Листинг 1. **Файл конфигурации**

```
const nconf = require("nconf");
const path =  require("path");
nconf.argv()
   .env()
   .file({file: path.join(__dirname, "config.json")});
module.exports = nconf;
```

Папка «public» хранит в себе публичную информацию, в основном это статика, готовая верста, изображения, таблицы стилей.

Папка «node_modules» хранит в себе все зависимости проекта, все библиотеки, которые были скачаны в проект, помещаются в данную папку.

Папка «controllers» будет содержать все контроллеры, и будет делиться по сущностям, то есть, если в проекте есть user, то в папке controlles будет папка user, которая хранит в себе все контроллеры для его обработки.

Папка «routes» хранит в себе соответствие контроллеров и промежуточных обработчиков для разных url адресов, в данной папке будет описано, на какие запросы какие контроллеры должны реагировать, а также будет валидация запроса и его документация. Структура папки будет также по сущностям.

Папка «libs» будет содержать какие-либо библиотеки, которые было необходимо создавать в ходе проекта. Данные библиотеки могут лишь принимать наборы параметров и не могут взаимодействовать с моделями проекта.

Папка «services» будет содержать сложную логику проекта, например, для переключения сценария в проекте не получится просто сменить тип у модуля, потому для данного действия будет написан сервис. Структура папки также по моделям.

Папка «logs» будет содержать файлы с логами сервера.

Папка «utils» будет хранить отдельно описанные наборы функций для работы с модулями, она будет являться той самой прослойкой между сервером и модулем.

Папка «models» хранит в себе данные обо всех моделях с их валидацией. А также собирательный файл, который проводит всю синхронизацию моделей с БД.

Папка «middlewares» хранит в себе промежуточные обработчики, те функции, которые будут вызываться перед запуском контроллера, например, в проекте всегда пишут middleware для авторизации пользователя и принятия решения, пускать его в этот контроллер или нет.

В файле «package.json» хранится конфигурационная информация для сервера, его зависимости (библиотеки) и скрипты для запуска.

В файле «index.js» будет храниться стартовая информация для запуска сервера. Такая, как создание приложения express, импорт библиотек, а также синхронизация моделей БД, если синхронизация не произойдет, то выпадет ошибка и сервер не запустится. Также будет проходить инициализация промежуточных функций, которые используются для всех запросов сервера. Делаться настройка CORS, чтобы запросы с любых URL адресов могли приходить и обрабатываться сервером, для того, чтобы система в дальнейшем могла интегрироваться с другими системами без вмешательства разработчика. В этом же файле будет подключение файла с роутингом, а

также настройка для раздачи статических файлов и в заключении будет настройка перенаправления и обработчик 404 ошибки.

3.6.2 Разработка моделей БД

Далее необходимо разработать модели БД, чтобы в дальнейшем можно было легко с ними взаимодействовать.

Для работы с БД будет использоваться ORM система «sequelize», которая позволяет работать с СУБД без использования SQL запросов, а за счет заранее описанных моделей данная библиотека самостоятельно составит SQL код и сделает необходимые запросы [20].

Сначала необходимо подключиться к БД. Это делается в папке «config/connect»

Далее создаются модели проекта, пример создания модели можно увидеть в листинге 2.

Листинг 2. Пример создания модели

```
const Sequelize = require('sequelize');
const sequelize = require('../config/connect');

class BuildingModel {
   constructor({name}){
      this.name = name;
   };
};

const Model = Sequelize.Model;

class Buildings extends Model { }

Buildings.init({
   name: {
      type: Sequelize.STRING,
      unique: true,
   }
}, {
   sequelize,
```

```
  modelName: 'buildings'
});

module.exports = {
  Buildings,
  BuildingModel,
};
```

Вверху происходит импорт библиотек, после создается класс модели, данный класс в дальнейшем будет использоваться для создания сущности, потому у него нет таких атрибутов, как «id», «updatedAt» и «createdAt».

Далее создается класс, который наследуется от класса «Model» из библиотеки sequelize. В методе «init» указывается список колонок таблицы, колонки «id», «updatedAt» и «createdAt» создаются автоматически. В дополнительных параметрах необходимо прописать «sequelize» - это объект подключения к БД и название модели, оно будет использоваться для создания БД.

Данные модели также помогают производить валидацию, как это показано в листинге 3.

Листинг 3. **Пример кода валидации**

```
  validate: {
    notNull: {
      msg: "Поле является обязательным!",
    },
    notEmpty: {
      msg: "Поле является обязательным!",
    },
    len: {
      args: [[1, 255]],
      msg: "Пароль должен быть от 1 до 255 символов!"
    },
  },,
```

Здесь сразу задается сообщение ошибки, а также описываются разные валидации и ошибки, которые будут выдаваться в этом случае.

3.6.3 Разработка контроллеров

Далее необходимо организовать работу сервера, для этого будут написаны контроллеры, они будут в дальнейшем обрабатывать запросы сервера [24].

Структура всех контроллеров будет иметь одинаковый подход в написании: импортирование сущностей, с которыми необходимо работать, далее происходит создание обработки запроса. Все контроллеры обернуты в блоки для обработки ошибок, все ошибки будут обрабатываться одной общей функцией для нормализации ошибок, также в этой функции будет происходить логирование ошибок.

Далее необходимо написать один из наиболее важных контроллеров, он показан в листинге 4.

Листинг 4. **Контроллер выставления сценария**

```
const { setScenario } = require("../../services/modules/setScenario");
const { errorHandler } = require("../../libs/errorHandler");

module.exports.setScenarioByModuleId = async function (req, res) {
  try {
    const newModule = await setScenario({
      moduleId: req.body.moduleId,
      scenarioId: req.body.scenarioId,
      getNew: true,
    });
    res.status(200).json({
      success: true,
      data: newModule
    });
  } catch (e) {
    console.log(e);
    errorHandler(e, res);
  }
};
```

Данный контроллер необходим для смены сценария у модуля, так как в дальнейшем данная логика может быть переиспользована, то было принято решение вынести ее в отдельный сервис. Весь этот сервис показан в листинге Л.1.

Здесь происходит поиск модуля и сценария по id, после чего данные модели отправляются в сторонний сервис, который сменяет состояние модуля, он принимает 2 аргумента: модуль и новый сценарий.

Далее идет определение даты смены сценария модуля, после чего данные записываются в БД и обновляется информация об ошибке модуля.

Далее происходит проверка, что если необходимо вернуть данный объект, то происходит его поиск в БД со всеми соответствующими присоединениями.

Если же новый объект возвращать не нужно, то возвращается пустой объект.

Также необходимо сделать контроллер для создания какой-либо сущности. Такой контроллер принимает тело запроса, которое ему было отправлено в JSON формате, после чего создает на его основе объект от соответствующего класса, который запрещает выставление некоторых дополнительных полей. Далее создается сущность через ORM.

По такому принципу происходит создание все контроллеров приложения.

3.6.4 Разработка аутентификации и авторизации

Аутентификация будет работать через контроллер «signIn», который будет проверять логин и пароль пользователя. Если пользователь с такими логином и паролем нашелся, то будет создаваться jwt токен для дальнейшей авторизации пользователя в системе.

Далее для каждого действия для аутентифицированного пользователя необходимо делать проверку, есть ли у пользователя необходимые права для данного действия или нет. Для этого был написан создатель промежуточного

обработчика, который делал эту проверку, пример его кода изображен в листинге М.1.

Данная функция принимает на вход список, массив уникальных идентификаторов ролей или же строковое значение «all», которое говорит, что доступ есть у всех, кто смог подтвердить свой токен.

Также была написана промежуточная функция обработки параметров запроса. Модели в проекте проходят валидацию самостоятельно, но, например, параметры get запроса не попадают, ни в какую модель, но их необходимо валидировать. Для этого была написана промежуточная функция. Данная функция принимает набор параметров для валидации и допустимые значения, если какой-то из параметров не проходит данную валидацию, то обработка запроса завершится на данном этапе и выдаст ошибку.

3.7 Разработка клиентской части АИС

3.7.1 Разработка структуры приложения

В первую очередь необходимо разработать структуру для клиентской части приложения.

Поскольку клиентская часть будет разрабатываться на React, то для создания проекта использовалась утилита CRA. А также был применен один из наиболее распространенных подходов к разработке структуры – это деление на контейнеры и компоненты.

Структура проекта представлена на рисунке Н.1.

Вся структура проекта помещается в папке frontend.

В папке «public» находится шаблон для работы с конечным React приложением, в данной папке можно поправить стандартный html документ приложения, если в этом будет необходимость.

В папке «node_modules» и файле «package.json» хранится аналогичная информация, как и в серверной части.

В папке «build» будет лежать конечный код React приложения, который будет состоять из абсолютно валидных HTML, CSS и JS.

В папке «src» будут находиться все исходные файлы приложения, которые необходимы для разработки.

В папке «api» будут находиться все запросы к серверу с описанием их параметров и возвращаемой информацией.

В папке «containers» будут храниться контейнеры – это React компоненты, которые занимаются расположением компонентов на основе данных. Также контейнеры являются компоновщиками компонентов.

В папке «assets» будет храниться статическая информация, такая, как шрифты, изображения проекта и общие стили проекта.

В папке «mobx» будет располагаться глобальное хранилище данных, с которым в дальнейшем будут работать контейнеры приложения.

В папке «components» будут храниться компоненты – это React компоненты, которые отвечают только за визуальное отображение на основе своих входных параметров и которые никак не взаимодействуют с приложением напрямую, но могут вызывать функции, переданные им сверху.

В папке «consts» будут находиться файлы со всеми константами проекта.

В папке «types» будет находиться список всех доступных типов, которые будут приходить с сервера или которые будут использоваться в приложении.

В папке «utils» будут находиться те сторонние функции, которые позволяют не дублировать один и тот же код, например функции для работы с cookies файлами.

В файле «index.tsx» будет находиться запуск React приложения.

3.7.2 Разработка работы с модулями

Далее необходимо разработать логику работы с модулями, для этого будет использоваться moduleStore, пример экрана показан на рисунке 4.

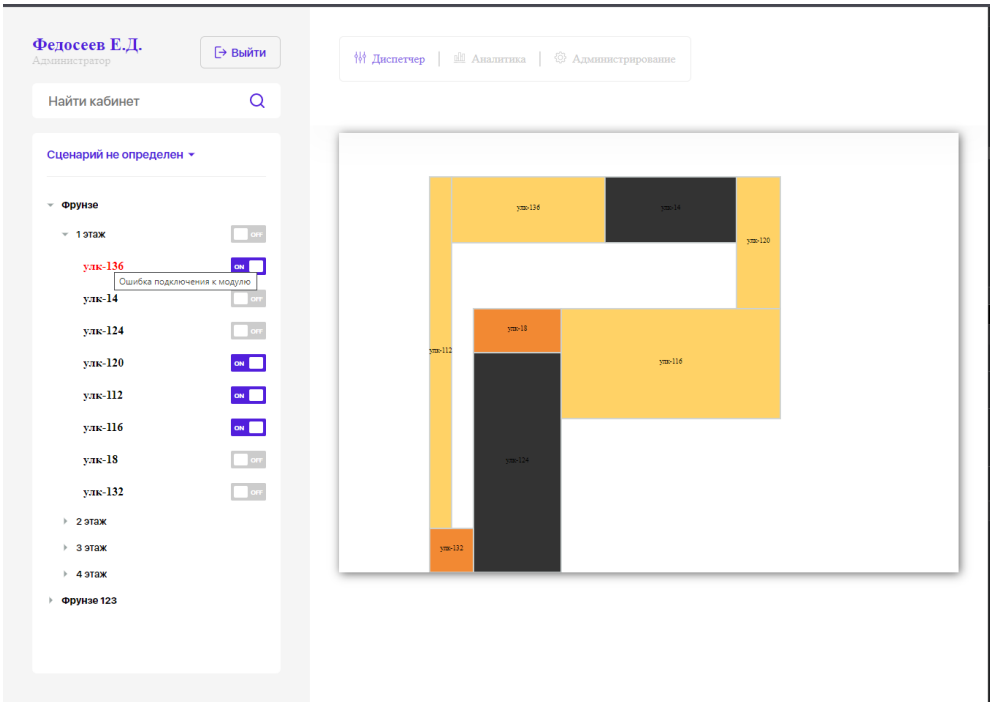

Рисунок 4 – Пример экрана главной страницы

Для такого отображения необходимо создать хранилище, в котором будут следующие данные:

- список всех зданий;
- список всех этажей для здания;
- список всех модулей для этажа;
- активный сценарий;
- id модуля, который был найден через поиск, чтобы открыть его;
- состояние открытых списков;
- состояние модуля:
  - его сценарий;
  - состояние в загрузке или нет.

Далее необходимо создать работу списка модулей, для этого необходимо создать компонент, как показано в листинге П.1.

Каждый модуль выглядит следующим образом, идет проверка на то, какой сценарий сейчас выбран, а также создается useEffect хук, который откроет данный модуль, если он был выбран в поиске. Далее в компоненте идет проверка на наличие прав у пользователя для смены сценария модуля, а также задается состояние для данных переключателей.

Далее необходимо разработать карту, код карты представлен в листинге Р.1.

Карта представляет из себя простое SVG изображение, которое дополнено JavaScript кодом для удобного взаимодействия с ней. Также на элементы SVG (модули) накладываются слушатели, которые также переключают состояние для модуля, для активации необходимо сделать двойной клик по тому модулю, состояние которого необходимо сменить.

3.7.3 Разработка форм

Далее необходимо разработать формы, для этого будет использоваться библиотека «Formik», она позволяет легко и просто работать с формами в React. На рисунке 5 показан пример формы, которую необходимо разработать [22].

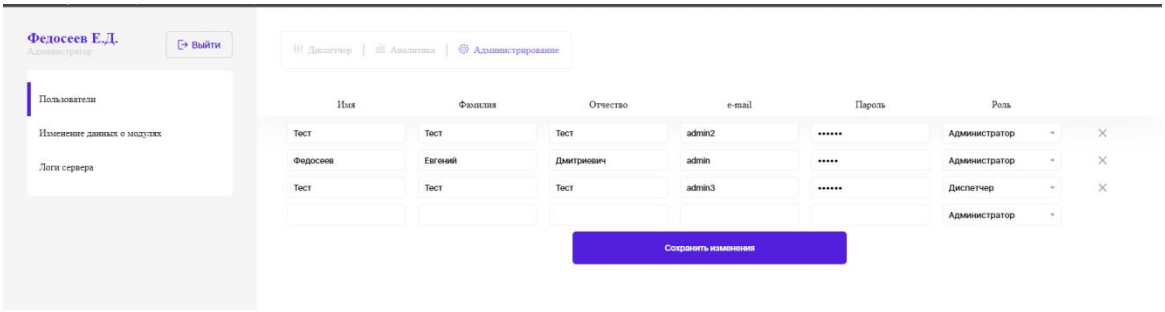

Рисунок 5 – Форма для редактирования пользователя

Также у данной формы должно быть всплывающее окно, как показано на рисунке 6.

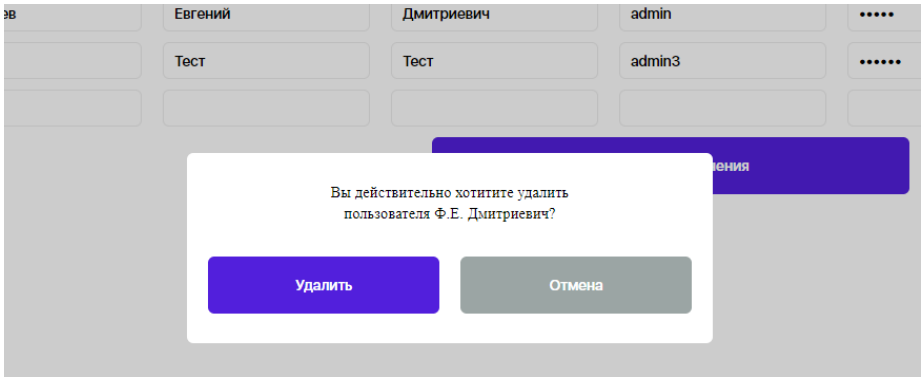

Рисунок 6 – Всплывающее окно подтверждения

Формы создаются по аналогии, показанной ниже. Для начала необходимо создать контекст от «Formik» и поместить туда разрабатываемую форму, как это показано в листинге 5.

Листинг 5. **Пример кода таблицы формы для пользователей**

```
return (
        {!!usersList.length && <Formik initialValues={initialValues}
onSubmit={onSubmit}>
          <UsersForm
            deleteUser={deleteItem}
            roles={rolesList}
          />
        </Formik>}
  );
```

В данном компоненте создается форма, выше описывается вся логика работы формы, например стандартные значения, обработка ошибок и обработка запроса.

Пример обработки запроса можно увидеть в листинге С.1. Данная функция для работы с пользователями позволяет пройти по всем пользователям в форме, после чего происходит проверка, если пользователь был добавлен в интерфейсе, тогда отправляется запрос на создание, если пользователь уже существовал, то осуществляется проверка на то, был ли пользователь отредактирован, если редактирование произошло, тогда

отправляется запрос на редактирование сущности. В случае, если какой-то запрос завершает выполнение с ошибкой, тогда его ошибка помещается в массив для ошибок и после обработки всех пользователей ошибки выводятся в форме, таким образом, получается, что в случае ошибок валидные данные для других пользователей будут сохранены на сервере.

Также необходимо получить исходные значения для формы пользователей, как показано в листинге Т.1. Здесь создается состояние для ролей и пользователей, после чего через хук «useCallback» создаются 2 запроса на сервер. После чего они вызываются в «useEffect», а во время вызова в приложении выставляется состояние загрузки.

Также необходимо разработать функцию для удаления пользователя из списка, так как это действие сопровождается дополнительной проверкой, то оно не нуждается в отправке формы.

Принцип работы функции следующий: идет проверка, что пользователь новый, то есть был создан только в интерфейс и не сохранен на сервере, тогда необходимо просто выйти из функции, а если все-таки нужно отправить запрос, тогда он отправляется.

В самой форме пользователя создаются все поля по аналогии с примером в листинге 6.

Листинг 6. **Поля формы**

```
<Col>
  <InputWithLabel
    mini
    onChange={handleChange}
    type="text"
    name={`data[${index}].surname`}
    value={value.surname}/>
</Col>
```

Также дополнительно создаются 2 функции и состояния для них, как показано в листинге У.1.

Здесь функция для удаления пользователя, которая берет пользователя из аргумента, удаляет его из своего массива, после чего вызывает функцию удаления из аргументов. А также функция для добавления пользователя. Функция добавления будет вызываться, когда пользователь полностью заполнит, необходимые поля формы.

При нажатии на кнопку удаления будет вызываться функция выставления пользователя для удаления, как это показано в листинге 7.

*Листинг 7.***Пример кода кнопки удаления**

```
<Col col={1} flexCenter>
  {index !== values.data.length - 1 && <CrossButton
    type="button"
    onClick={() => setUserForDelete(value)}/>}
</Col>
```

А внизу создается всплывающее окно, которое отображается пользователю, если есть пользователь на удаление.

Таким образом, была полностью реализована вся работа с формами и в других компонентах программы.

3.8 Разработка автоматической выгрузки проекта

Для настройки автоматической выгрузки необходимо написать скрипт, который будет выполнять следующие действия:

- скачивать библиотеки для серверной части;
- скачивать библиотеки для клиентской части;
- собирать клиентскую часть;
- перемещать собранную клиентскую часть в папку для статики сервера;
- производить выгрузку проекта на хостинг.

Так как для хранения исходного кода проекта используется web-сервис «gitlab», то на данном сервисе есть возможность написания скриптов, которые будут запускаться при каждом изменении проекта на этом сервисе. Для того, чтобы скрипты работали необходимо создать файл «.gitlab-ci.yml». Содержимое этого файла должно выполнять описанные выше действия, для этого был написан следующий код, показанный в листинге Ф.1.

В данном коде сказано, что необходим контейнер с конкретной версией Node.js. В предварительных скриптах необходимо скачать библиотеки, которые будут осуществлять выгрузку проекта на сервис heroku. Далее происходит скачивание зависимостей и сборка клиентской части, после чего происходит сама выгрузка проекта на сервис. Выгрузка происходит на основе ключей, которые лежат в конфигурации репозитория.

3.9 Тестирование

3.9.1 Тестирование кроссбраузерности АИС

Для проверки работы сайта в различных браузерах, необходимо провести кроссбраузерное тестирование.

Из требований к информационной системе можно сделать вывод, что сайт должен корректно отображаться в следующих браузерах последней версии:

- Mozilla Firefox,
- Google Chrome,
- Яндекс.Браузер.

Для того, чтобы проверить работу кроссбраузерности необходимо выдвинуть список тест-кейсов, который должен выполняться во всех, описанных выше, браузерах: Такой список будет представлен в таблице Х.1.

Все тест-кейсы были проведены в описанных выше браузерах и с успехом выполнены информационной системой. Следовательно, можно

сделать выводы, что разработанная информационная система корректно работает во всех, описанных выше, браузерах на разрешении 1920*1080 пикселей, а значит, удовлетворяет техническим требованиям.

3.9.2 Функциональное тестирование

Для того, чтобы произвести функциональное тестирование необходимо выделить ряд кейсов, которые необходимо проверить, так как основной функциональной частью разработанной информационной системы является работа с модулями, то тестирование будет производиться именно на них, для этого был составлен список тест-кейсов, который показан в таблице Э.1.

Все тест-кейсы, описанные в приложении выше, были успешно пройдены исходя из чего можно сделать вывод, что разрабатываемая информационная система корректно работает и полностью удовлетворяет выдвинутым техническим требованиям.

Выводы по главе 3

В результате данной главы была разработана информационная система по управлению «умным» освещением. Разработанная информационная система полностью удовлетворяет всем выдвинутым требованиям.

Система была протестирована и не показа никаких ошибок при ее использовании.

Заключение

В процессе выполнения выпускной квалификационной работы была проанализирована АХО ТГУ. Была выделена актуальность исследуемой темы, определены объект, предмет исследования, цели и задачи работы. Данный анализ показал, какие бизнес-процессы необходимо оптимизировать.

Были выявлены требования, на основе которых можно было бы сравнить аналоги. Был проведен сравнительный анализ существующих аналогов. Он показал, что существующие аналоги на данный момент не поддерживают большинство требований, которые были выдвинуты к разрабатываемой ИС.

Была разработана диаграмма компонентов и логическая модель данных для информационной системы.

Была проведена разработка информационной системы, в которую входило:

- выбор архитектуры и технологий для реализации,
- выбор СУБД и разработка физической модели данных,
- написание программного кода,
- тестирование.

Результатом выпускной квалификационной работы является разработанная информационная система автоматизации управления освещением в кампусе ТГУ, которая позволяет:

- управлять освещением,
- собирать статистику,
- проводить менеджмент пользователей,
- проводить менеджмент модулей,
- просматривать логи сервера.

После проведения тестирования можно сделать вывод, что разработанная информационная система не имеет полностью нерабочих модулей и удовлетворяет поставленным выше требования.

Список используемой литературы

1. База данных SQl или NoSQL: какую выбрать для проекта? [Электронный ресурс]: Сайт со сравнительным анализом SQL и NoSQL БД. URL: https://mcs.mail.ru/blog/sravnenie-sql-i-nosql-kak-vybrat-sistemu-hraneniya-dannyh (Дата обращения: 20.03.2021)
2. Диаграмма прецедентов (вариантов использования) UML | Планерка: коучинг профессиональной самореализации и креативности [Электронный ресурс]: Сайт с подробным описанием составления диаграммы прецедентов. URL: https://planerka.info/item/diagramma-precedentov-variantov-ispolzovaniya-uml/ (Дата обращения: 20.01.2021)
3. Емельянова, Н.З. Проектирование информационных систем: учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. – М.:Форум, 2014.
4. Изучаем PostgreSQL 10: учебное пособие / Джуба Салахалдин, Волков Андрей, 2019.
5. Концептуальное проектирование с использованием методологии IDEF1X - Учебная и научная деятельность Анисимова Владимира Викторовича [Электронный ресурс]: Сайт с описание методологии IDEF1X. URL: https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema7/tema7_2 (Дата обращения: 01.03.2021)
6. Основы методологии IDEF1X [Электронный ресурс]: Сайт с описанием методологии IDEF1X. URL: https://www.cfin.ru/vernikov/idef/idef1x.shtml (Дата обращения: 02.03.2021)
7. Перечень услуг, оказываемых студентам и сотрудникам ТГУ в рамках службы поддержки административно-хозяйственного обслуживания [Электронный ресурс]: Документ с описание услуг АХО. URL:

http://edu.tltsu.ru/sites/sites_content/site117/html/media84888/22_Pere4en_22032021.pdf (Дата обращения: 24.12.2021)

8. Простое руководство по UML-диаграммам и моделированию баз данных [Электронный ресурс]: Сайт с преимуществами и типами UML-диаграмм. URL: https://www.microsoft.com/ru-ru/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (Дата обращения: 02.01.2021)
9. Сравнение MySQL и PostgreSQL | Losst [Электронный ресурс]: Сайт со сравнением MySQL и PostgreSQL. URL: https://losst.ru/sravnenie-mysql-i-postgresql#Хранение_данных (Дата обращения: 15.03.2021)
10. Сравнительный анализ нотаций ARIS/IDEF | iteam [Электронный ресурс]: Сайт с описание нотации ARIS. URL: https://blog.iteam.ru/sravnitelnyj-analiz-notatsij-aris-idef-i-produktov-ih-podderzhivayushhih-aris-toolset-bpwin/ (Дата обращения: 10.01.2021)
11. Требования к системе: классификация FURPS+ — SysAna [Электронный ресурс] URL: https://sysana.wordpress.com/2010/09/16/furps/ (Дата обращения: 01.02.2021)
12. ФСИС_семинар-9_Диаграмма-компонентов [Электронный ресурс]: Документ с примерами диаграмм компонентов. URL: http://imlearning.ru/netcat_files/file/FSIS/ФСИС_семинар-9_Диаграмма-компонентов.pdf (Дата обращения: 01.03.2021)
13. SADT [Электронный ресурс]: Сайт с подробным описанием методологии SADT. URL: http://or-rsv.narod.ru/SADT/SADT.htm (Дата обращения: 10.01.2021)
14. {Вы не знаете JS} Замыкания и объекты: учебное пособие / Кайл Симпсон, 2019.
15. UML.book [Электронный ресурс]: Сайт с подробным описанием методологии UML. URL:

https://picloud.pw/media/resources/posts/2018/02/20/UML_основы.pdf (Дата обращения: 20.01.2021)

16. AngularJS: Tutorial: Tutorial [Электронный ресурс]: Сайт с документацией по фреймворку AngularJS. URL: https://docs.angularjs.org/tutorial (Дата обращения: 25.03.2021)
17. ASP.NET documentation | Microsoft Docs [Электронный ресурс]: Сайт с документацией по ASP.NET. URL: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 (Дата обращения: 26.03.2021)
18. Getting Started – React [Электронный ресурс]: Сайт с документацией по фреймворку React. URL: https://reactjs.org/docs/getting-started.html (Дата обращения: 01.04.2021)
19. Introduction — Vue.js [Электронный ресурс]: Сайт с документацией по фреймворку Vue.js. URL: https://vuejs.org/v2/guide/ (Дата обращения: 01.04.2021)
20. Manual | Sequelize [Электронный ресурс]: Сайт с документацией по использованию ORM Sequelize. URL: https://sequelize.org/master/ (Дата обращения: 20.04.2021)
21. Node.js [Электронный ресурс]: Сайт Node.js. URL: https://nodejs.org/ru/ (Дата обращения: 01.04.2021)
22. Overview | Formik | Formik [Электронный ресурс]: Сайт с документацией к библиотеке Formik. URL: https://formik.org/docs/overview (Дата обращения: 21.04.2021)
23. Stack Overflow Developer Survey 2020 [Электронный ресурс]: Сайт со статистикой опроса программистов. URL: https://insights.stackoverflow.com/survey/2020#most-popular-technologies (Дата обращения: 20.03.2021)
24. Using Express middleware [Электронный ресурс]: Сайт с документацией по фремворку Express. URL:

https://expressjs.com/en/guide/using-middleware.html (Дата обращения: 20.04.2021)

Приложение А

Диаграмма прецедентов «КАК ЕСТЬ»

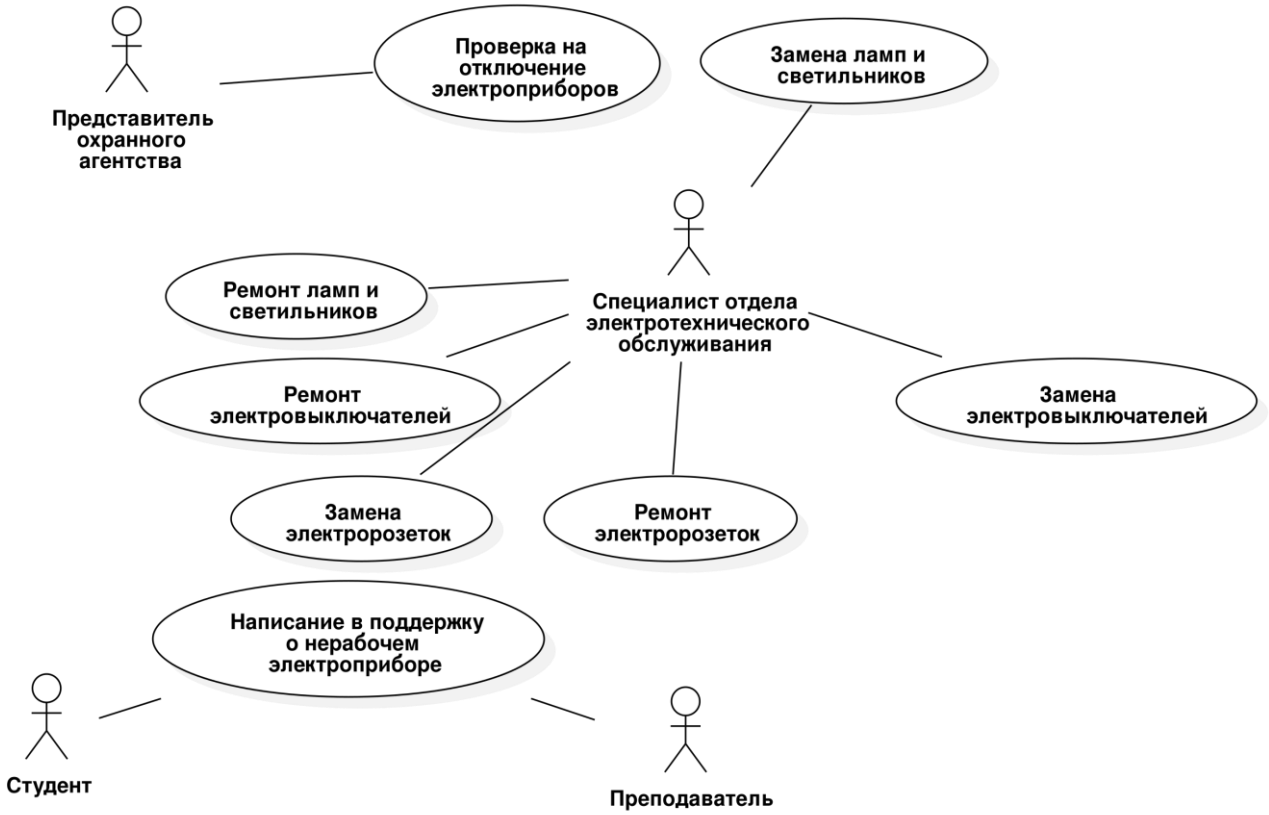

Рисунок А.1 – Диаграмма прецедентов «КАК ЕСТЬ»

Приложение Б

Диаграмма прецедентов «КАК ДОЛЖНО БЫТЬ»

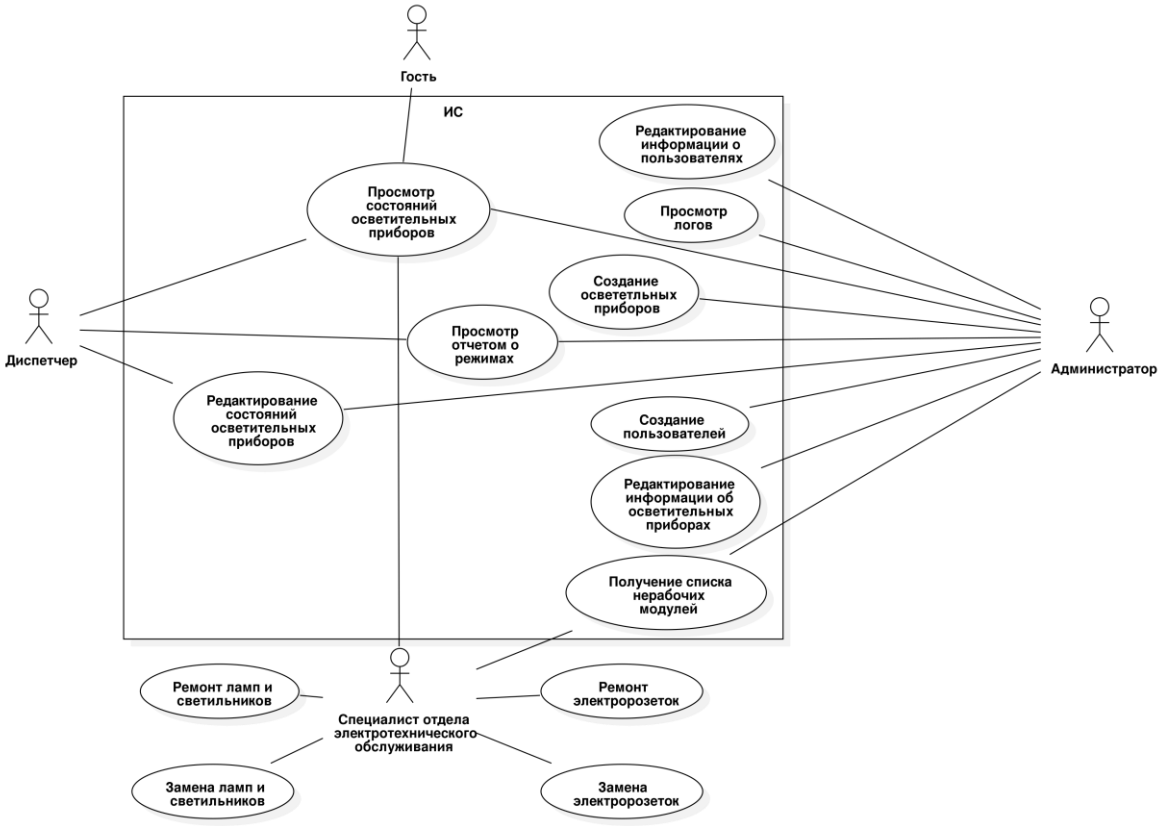

Рисунок Б.1 – Диаграмма прецедентов «КАК ДОЛЖНО БЫТЬ»

Приложение В

Диаграмма видов деятельности для смены состояния модуля

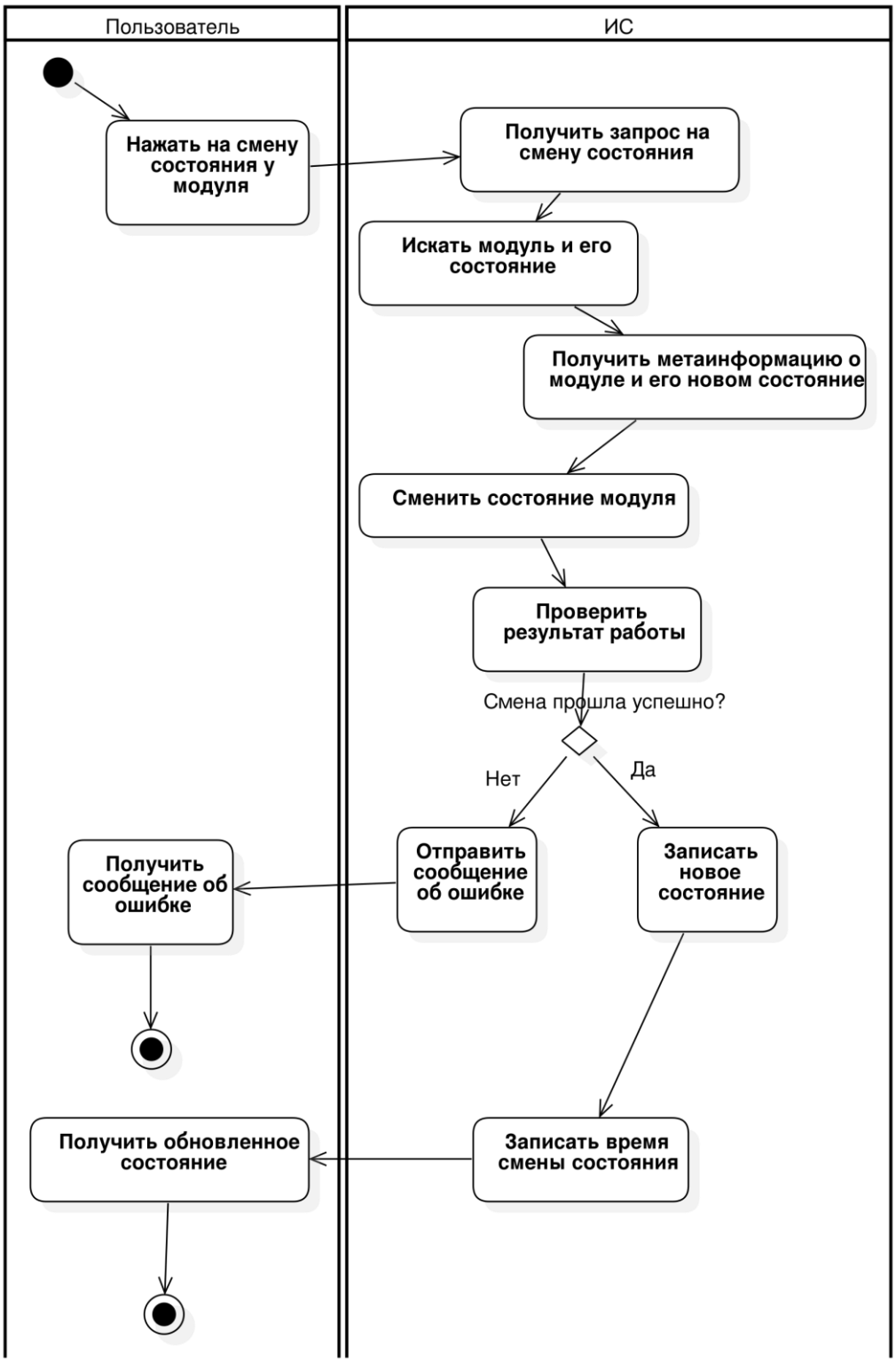

Рисунок В.1 – Диаграмма видов деятельности для смены состояния модуля

Приложение Г

Диаграмма компонентов

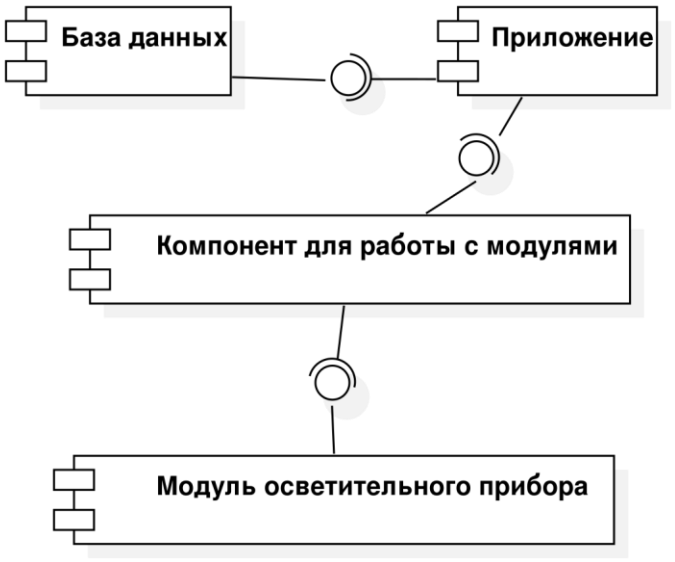

Рисунок Г.1 – Диаграмма компонентов

Приложение Д

Диаграмма последовательности для процесса смены состояния модуля

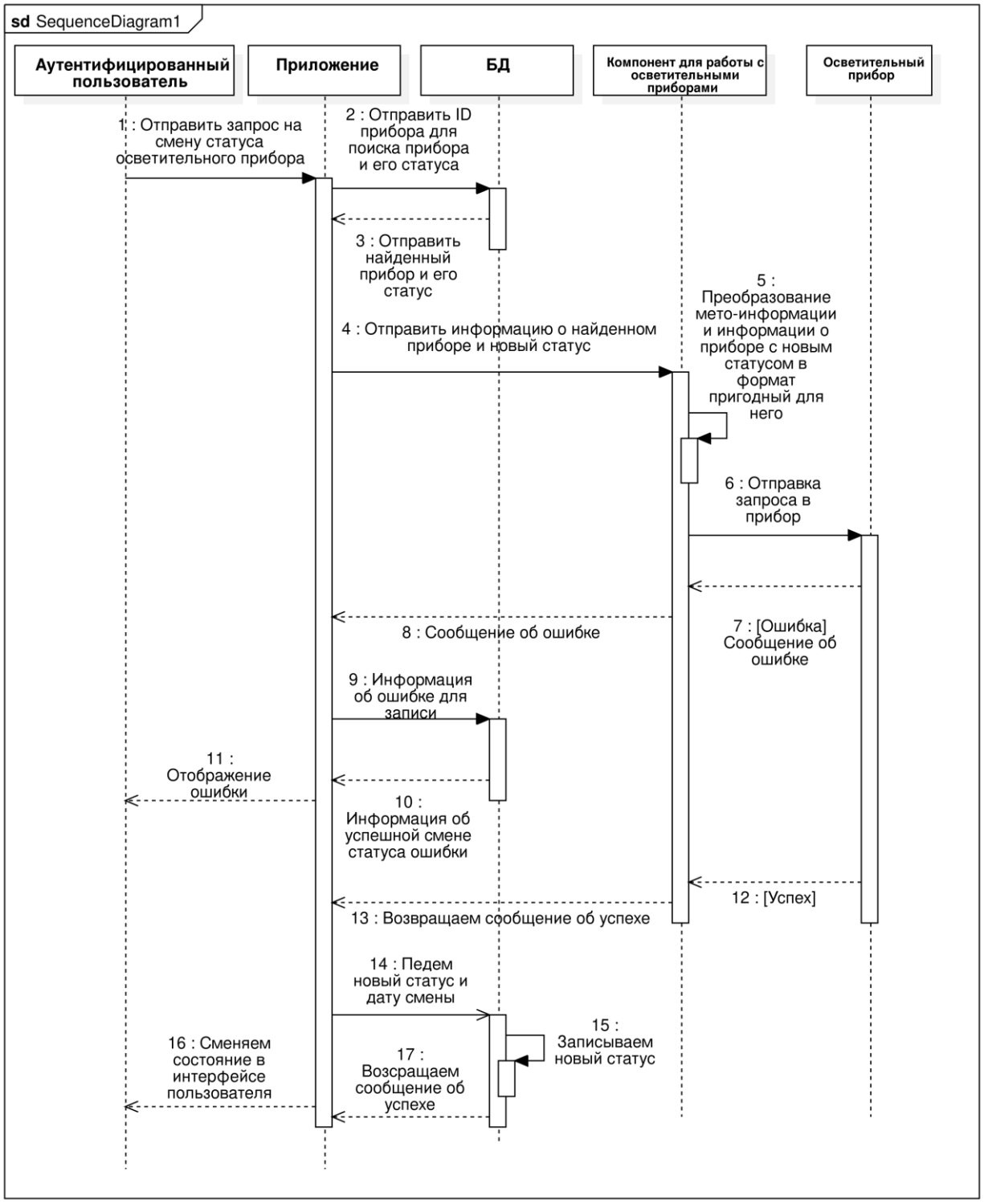

Рисунок Д.1 – Диаграмма последовательности для процесса смены состояния модуля

Приложение Е

Логическая модель БД

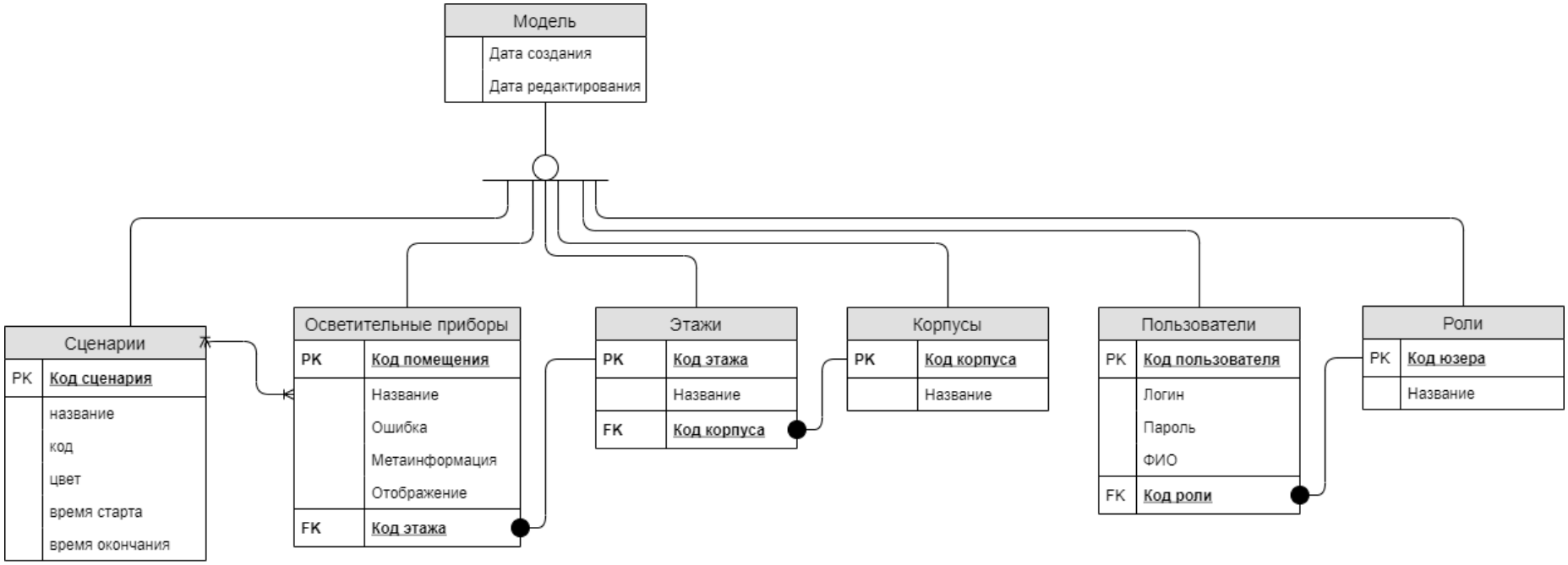

Рисунок Е.1 – Логическая модель БД

Приложение Ж

Физическая модель БД

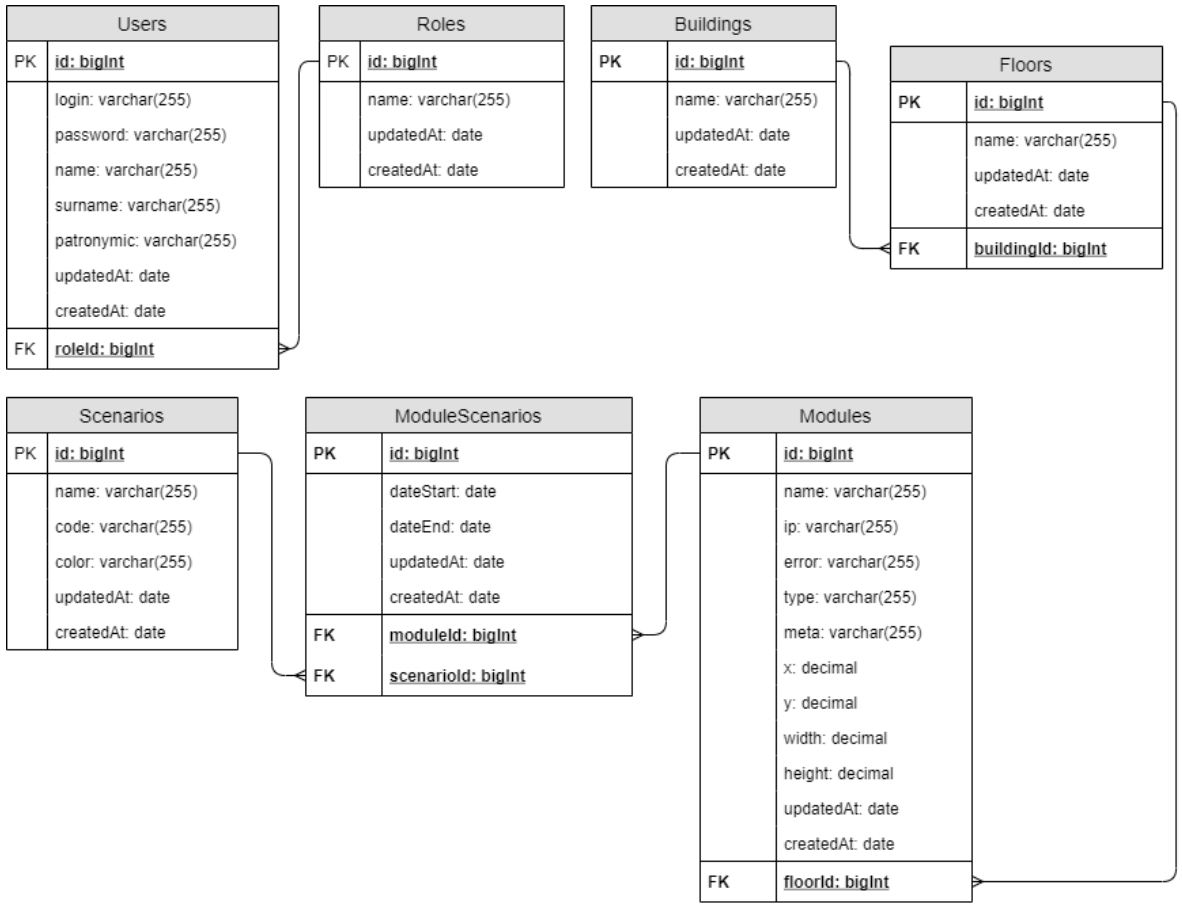

Рисунок Ж.1 – Физическая модель БД

Приложение И

Диаграмма развертывания

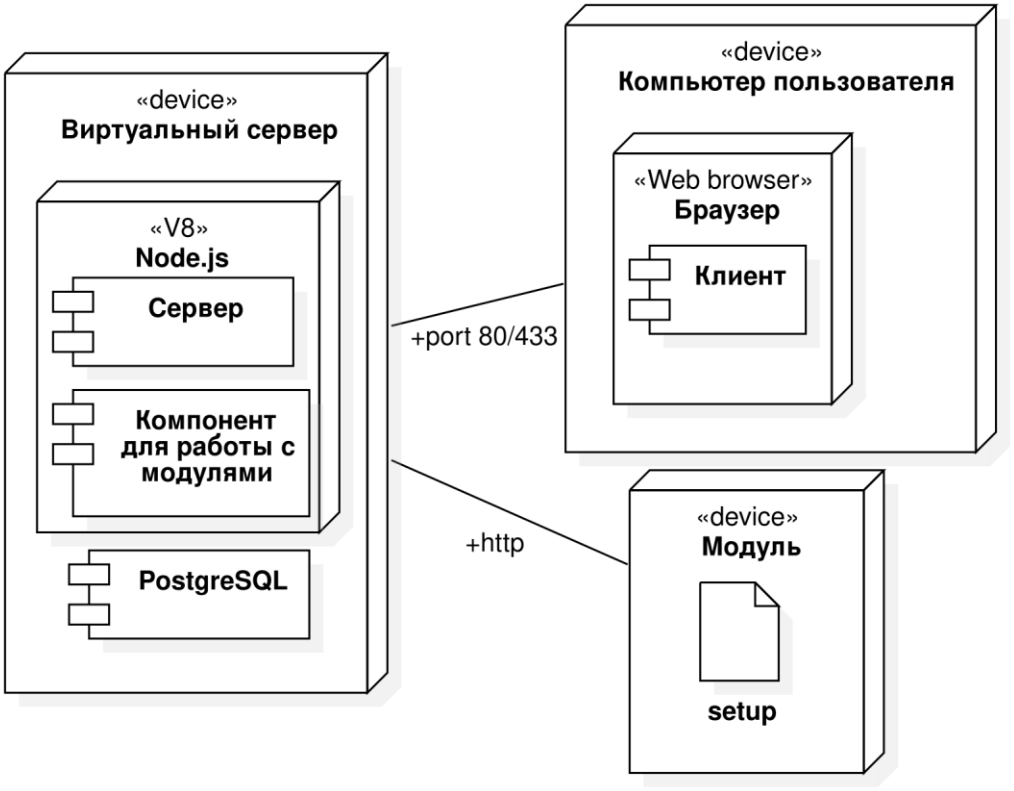

Рисунок И.1 – Диаграмма развертывания

Приложение К

Структура серверной части

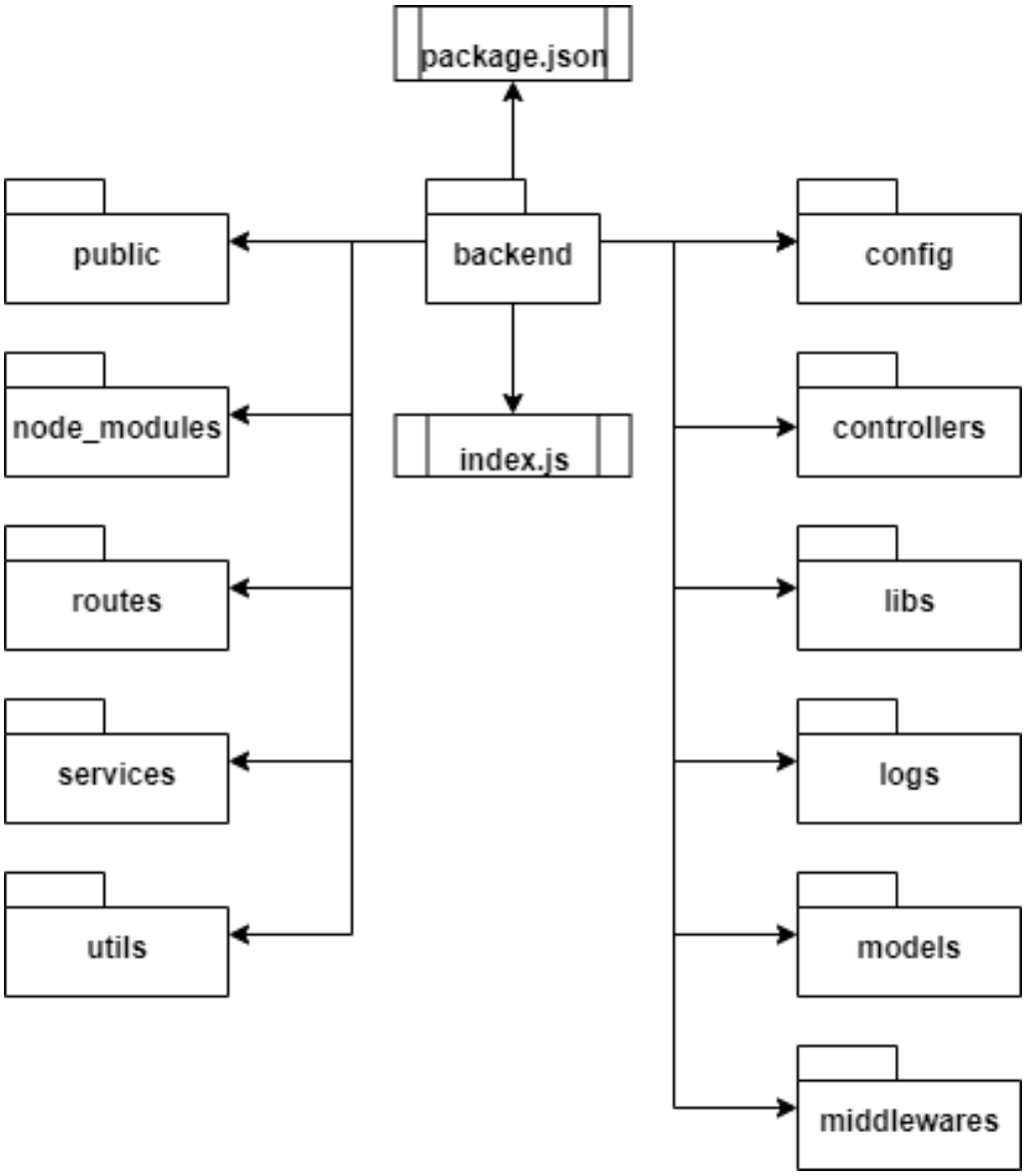

Рисунок К.1 – Структура серверной части

Приложение Л

Пример кода сервиса для выставления активного сценария

Листинг Л.1. **Пример кода сервиса для выставления активного сценария**

```
module.exports.setScenario = async function ({moduleId, scenarioId, getNew}) {
 try {
  const [_module, _scenario] = await Promise.all([
   await Modules.findOne({
    where: {id: moduleId}
   }),
   Scenarios.findOne({
    where: {id: scenarioId}
   })
  ]);

  if (!_module || !_scenario) {
   throw new Error('Ошибка!');
  }
  await setScenarioService({
   module: _module,
   scenario: _scenario,
  });

  const oldScenario = await ModuleScenarios.findOne({
   where: {
    dateEnd: null,
    moduleId: moduleId,
   },
  });

  const date = new Date();
  await oldScenario.update({dateEnd: date});

  await ModuleScenarios.create(new ModuleScenarioModel({
   moduleId: moduleId,
   scenarioId: scenarioId,
   dateEnd: null,
   dateStart: date,
  }));
```

```
    await Modules.update({
      error: null,
    }, {
      where: {
        id: moduleId,
      }
    });
    if (getNew) {
      const newModule = await Modules.findOne({
        where: {id: moduleId},
        include: {
          model: ModuleScenarios,
          where: {
            [Sequelize.Op.and]: [
              {dateEnd: null},
            ]
          },
          include: {
            model: Scenarios,
          }
        },
      });

      return newModule;
    } else {
      return {};
    }

  } catch (e) {
    if(e.errorText) {
      await Modules.update({error: e.errorText}, {
        where: {id: moduleId}
      })
    }
    throw e;
  }
};
```

Приложение М

Пример кода промежуточной функции авторизации

Листинг М.1. **Пример кода промежуточной функции авторизации**

```
module.exports.verification = (...array) => async function (req, res, next) {
  try {
    jwt.verify(req.get("token"), jwcConf.secretOrKey, async function (err,
decoded) {
        if (decoded === undefined) {
          return res.status(401).json({
            success: false,
            data: 'Вы не авторизироавнны!',
          });
        const user = await Users.findOne({
          where: {id: decoded.id, },
        });
        if (!user) {
          res.status(401).json({
            success: false,
            data: 'Вы не авторизироавнны!',
          });
        } else {
          if(array.includes("all")) {
            return next();
          }
          if (array.includes(user.roleId)) {
            req.user = user;
            return next();}
          return res.status(403).json({
            success: false,
            data: 'Недостаточно прав!',
          });
        }
    })
  } catch (e) {
    console.log(e);
    return res.status(401).json({
      success: false,
      data: 'Вы не авторизироавнны!',
    })   }};
```

Приложение Н

Структура клиентской части

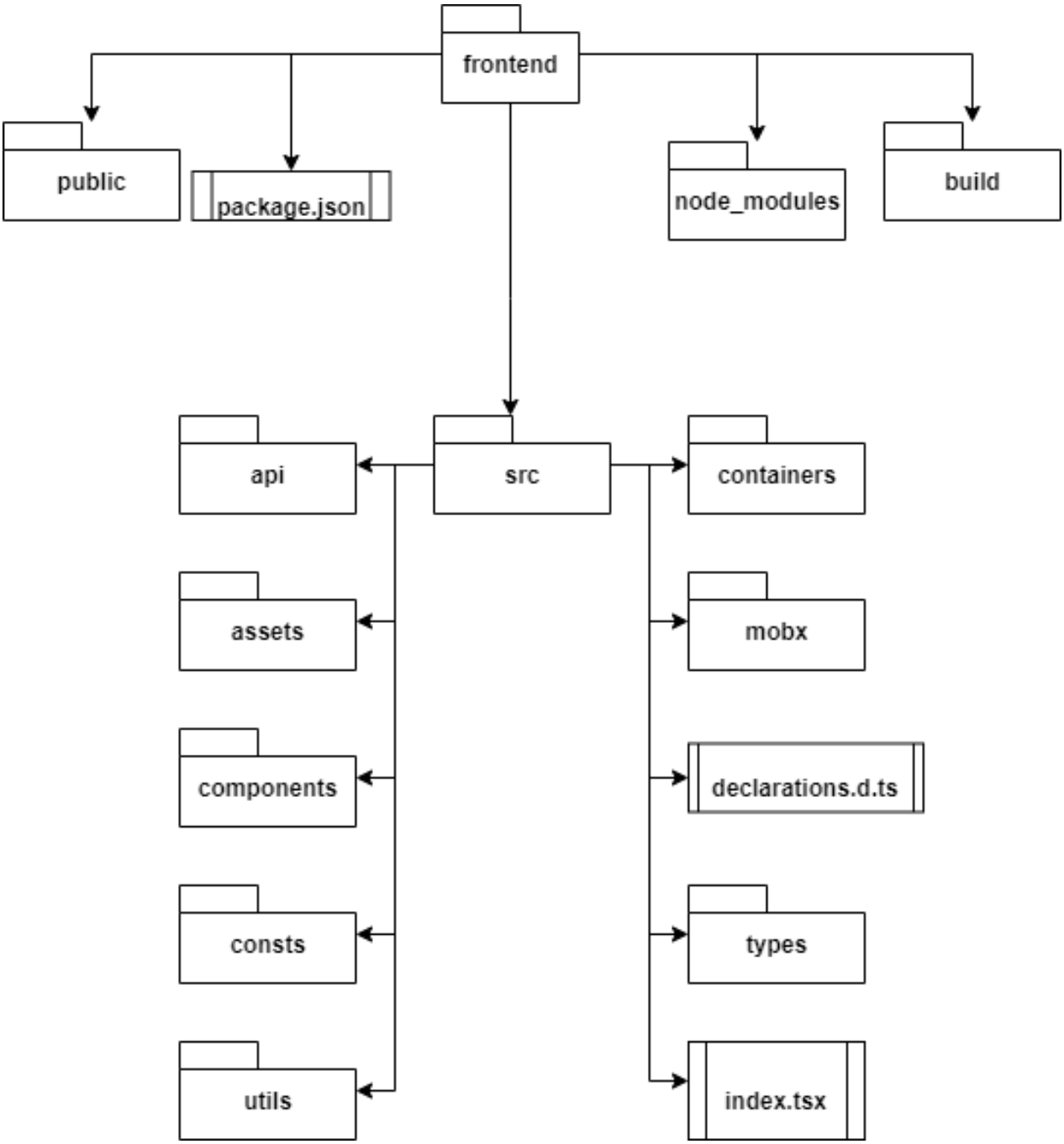

Рисунок Н.1 – Структура клиентской части

Приложение П

Пример кода функции «ModuleItem»

Листинг П.1. **Пример кода функции «ModuleItem»**

```
function ModuleItem({ error, withScenario, title, isActive , isLoading, onSwitch,
moduleRef, id }: ModuleItemProps) {
   const { modulesStore, authStore } = useContext(StoreContext);
   const { currentModuleId } = modulesStore;
   const role = authStore.userInfo.roleId;
   useEffect(() => {
      if (currentModuleId && currentModuleId === id) {
         setTimeout(() => {
            if (moduleRef.current) {
               moduleRef.current.scrollIntoView({ block: 'end', inline: 'nearest',
behavior: 'smooth' });
               modulesStore.resetCurrentModuleId();
            }
         }, 500);
      }
   }, [id, moduleRef, modulesStore, currentModuleId]);
   return(
      <li title={error || ''} className={classNames(
         'module-item',
         {'error-message-text': error}
      )} ref={moduleRef}>
         <div className='module-item__item'>
            {
               withScenario && <CheckButton disabled={role === 3}
isActive={isActive} onSwitch={onSwitch}/>
            }
            <h5 className={classNames('text-wrapper')}>{title}</h5>
         </div>
         {
            !withScenario && <SwitchButton disabled={role === 3}
isActive={isActive} onSwitch={onSwitch}/>
         }  {isLoading && <ControlPanelPreloader/>}
      </li>
   );
}
export default observer(ModuleItem);
```

Приложение Р

Пример кода карты с главной страницы

Листинг Р.1. **Пример кода карты с главной страницы**

```
function MainPage() {
   const {modulesStore, authStore} = useContext(StoreContext);
   async function onSwitch(moduleId: number) {
      await modulesStore.onModuleSwitch(moduleId);
   }
   const role = authStore.userInfo.roleId;
   return (
      <Body><> </>
         <>
            {!!modulesStore?.buildingsList?.find(({isOpen}) =>
isOpen)?.floorsList?.find(({isOpen}) => isOpen) &&
            <SVGViewer className='main-map' height={1000} width={1000}>
               {modulesStore?.buildingsList?.find(({isOpen}) =>
isOpen)?.floorsList?.find(({isOpen}) => isOpen)?.modulesList.map(module => {
                  return <>
                     <rect
                     onDoubleClick={() => role !== 3 && onSwitch(module.id)}
                     key={module.id} x={module.x} y={module.y}
width={module.width}
                     height={module.height}
                     style={{
                        'fill': module.moduleScenarios[0].scenario.color,
                        'stroke': '#CDD3D3',
                        'strokeWidth': '3',
                        'fillOpacity': 0.8,
                     }}/>
                     <text x={+module.x + module.width / 2} y={+module.y +
module.height / 2} style={{textAnchor:'middle', userSelect:
'none'}}>{module.name}</text>
                     </>
               })}
            </SVGViewer>}
         </>
      </Body>
   );}
export default observer(MainPage);
```

Приложение С

Пример кода обработки отправки формы

Листинг С.1. **Пример кода обработки отправки формы**

```
   async function onSubmit(values: UsersFormValues, {setErrors, setFieldValue}: {
setErrors: (errors: object) => void, setFieldValue: any }) {
      let errors = [];
      for (let index in values.data.slice(0, -1)) {
         const user = values.data[index];
         if (user.isNew) {
            loaderStore.setLoader(USERS_SUBMIT_LOADER);
            try {
               const response = await createUser(user);
               setFieldValue(`data[${index}]`, response.data);
               setUsersList(prevState => prevState.map(item => item.id === user.id ?
response.data : item));
               errors.push({});
            } catch (error) {
               errors.push(error.errors);
               if(error._message) {
                  errorMessageStore.showMessage(getErrorMessage(error));}
            } finally {
               loaderStore.removeLoader(USERS_SUBMIT_LOADER);
            }
         } else {
            const initialUser = usersList.find(({id}) => id === user.id);
            if      (initialUser      &&      !BOOLEAN.isObjectsEquals<FormUser>(user,
initialUser)) {
               loaderStore.setLoader(USERS_SUBMIT_LOADER);
               try {
                  const response = await editUserById(user);
                  setFieldValue(`data[${index}]`, response.data);
                  setUsersList(prevState => prevState.map(item => item.id === user.id
? response.data : item));
                  errors.push({});
               } catch (error) {
                  errors.push(error.errors);
                  if(error._message) {
                     errorMessageStore.showMessage(getErrorMessage(error));
                  }
```

```
            } finally {
              loaderStore.removeLoader(USERS_SUBMIT_LOADER);
            }
          } else {
            errors.push({});
          }
        }
      }
      setErrors({data: errors});
    }
```

Приложение Т

Пример кода для получения пользователя в форму

Листинг Т.1. **Пример кода для получения пользователя в форму**

```
const {loaderStore, errorMessageStore} = useContext(StoreContext);

const [usersList, setUsersList] = useState<Array<FormUser>>([]);
const [rolesList, setRolesList] = useState<Array<Role>>([]);

const getUsersList = useCallback(async () => {
   try {
      const response = await getAllUsers();
      setUsersList(formatUsersResponse(response.data));
   } catch (error) {
      errorMessageStore.showMessage(getErrorMessage(error));
   }
}, [errorMessageStore]);

const getRolesList = useCallback(async () => {
   try {
      const response = await getAllRoles();
      setRolesList(response.data);
   } catch (error) {
      errorMessageStore.showMessage(getErrorMessage(error));
   }
}, [errorMessageStore]);

useEffect(() => {

   async function initData() {
      loaderStore.setLoader(USERS_LIST_LOADER);
      await getUsersList();
      await getRolesList();
      loaderStore.removeLoader(USERS_LIST_LOADER);
   }

   initData().then();
}, [getRolesList, getUsersList, loaderStore]);
```

Приложение У

Пример кода для обработки удаления пользователя

Листинг У.1. **Пример кода для обработки удаления пользователя**

```
const {errorMessageStore} = useContext(StoreContext);

const {values, handleChange, setFieldValue} =
useFormikContext<UsersFormValues>();
const [userForDelete, setUserForDelete] = useState<FormUser | null>(null);

function onAddUser(event: FormEvent<HTMLFormElement>) {
   setFieldValue('data', [...values.data, new FormUser(undefined, true)]);
   handleChange(event);
}

async function onDeleteUser(user: FormUser) {
   try {
      await deleteUser(user);
      setFieldValue('data', values.data.filter(({id}) => id !== user.id));
   } catch (e) {
      if(e._message) {
         errorMessageStore.showMessage(getErrorMessage(e));
      }
   }
}
```

Приложение Ф

Код файла «.gitlab-ci.yml»

Листинг Ф.1. **Код файла «.gitlab-ci.yml»**

```
image: node:14.15.1-buster

before_script:
  - apt-get update -qy
  - apt-get install -y ruby-dev
  - gem install dpl

deploy:
  script:
    - cd frontend
    - yarn
    - yarn build
    - cd ..
    - cd backend
    - yarn
    - rm public/build -r
    - mv ../frontend/build public/build
    - cd ..
    - dpl --provider=heroku --app=$HEROKU_APP_PRODUCTION --api-
key=$HEROKU_API_KEY --skip_cleanup
  only:
    - master
```

Приложение Х

Тест-кейсы для кроссбраузерного тестирования

Таблица Х.1 – Тест-кейсы для кроссбраузерного тестирования

Название	Инструкции	Ожидаемый результат
Корректное отображение страницы входа	Открыть приложение от лица не аутентифицированного пользователя	Приложение недоступно. Форма входа влезает в экран и выглядит, как это показано в приложении Ц рисунок 1
Корректная работа карты на компьютерной версии	Открыть приложение от лица аутентифицированного пользователя, открыть главную страницу, сменить масштаб	Карта имеет возможность изменения масштаба
Корректная работа страницы «Аналитика»	Открыть приложение от лица аутентифицированного пользователя, у которого есть доступ к просмотру графиков, открыть страницу «Аналитика»	Графики отображаются во всех браузерах, пример отображения графика представлен в приложении Ш рисунок 1
Корректная работа форм на странице «Администрирование»	Открыть приложение от лица аутентифицированного пользователя с ролью администратора, открыть страницу редактирования аккаунтов	Все формы отображаются корректно и выводят ошибки для пользователя, пример вывода ошибок показан в приложении Щ рисунок 1
Корректная работа страницы с логами	Открыть приложение от лица аутентифицированного пользователя с ролью администратора, открыть страницу просмотра логов	Все логии доступны для скачивания и клик начинает скачивание
Корректная работа поиска	Открыть приложение от лица аутентифицированного пользователя, открыть главную страницу, ввести в поиске запрос, выбрать модуль	На главной странице найденный элемент должен сразу же открываться в выпадающем списке модулей

Приложение Ц

Пример отображения корректной страницы входа

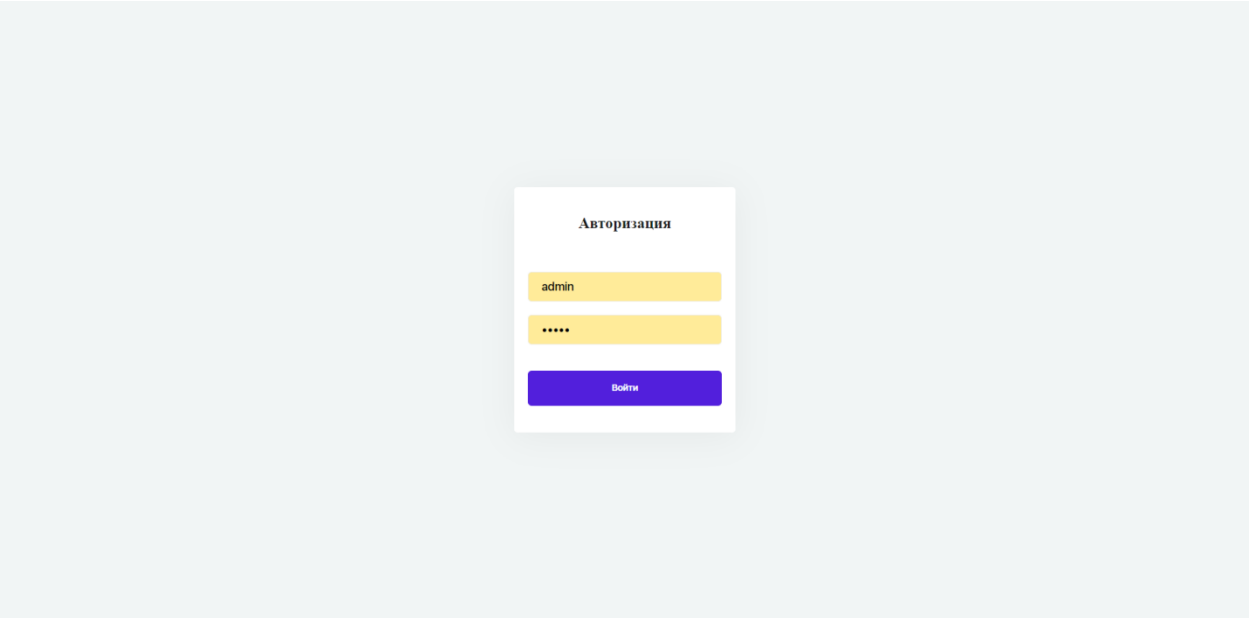

Рисунок Ц.1 – Пример отображения корректной страницы

Приложение Ш

Пример отображения графика

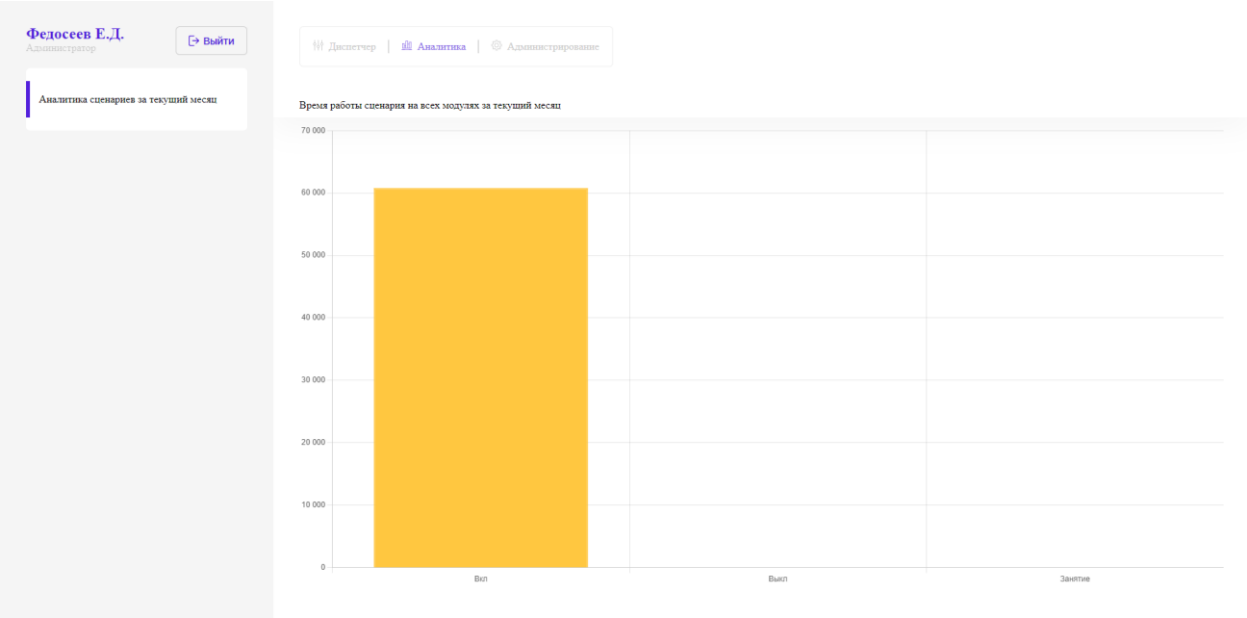

Рисунок Ш.1 – Пример отображения графика

Приложение Щ

Пример вывода ошибок для пользователя

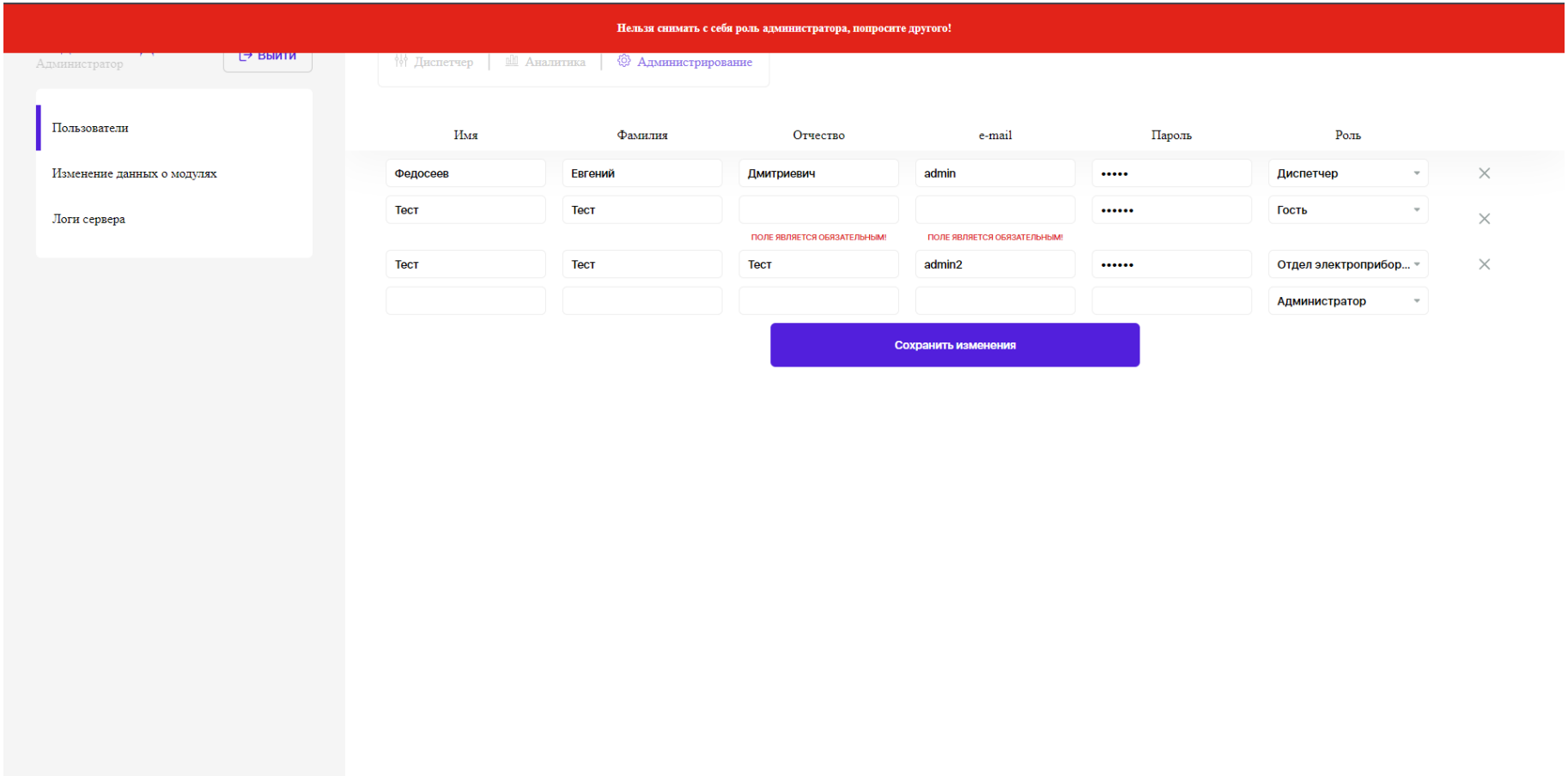

Рисунок Щ.1 – Пример вывода ошибок для пользователя

Приложение Э

Тест-кейсы для функционального тестирования

Таблица Э.1 – Тест-кейсы для функционального тестирования

Название	Инструкции	Ожидаемый результат
Поиск	Открыть приложение от лица аутентифицированного пользователя с правами на смену сценария у модуля. Вписать в поисковую строку на главной странице название модуля, сменить его сценарий	Модуль корректно сменил сценарий
Редактирование своего аккаунта	Открыть приложение от лица аутентифицированного пользователя с ролью администратора. Открыть страницу «Администрирование», найти в списке пользователя, от лица которого происходит сессия, сменить свои данные, нажать кнопку «Сохранить изменения».	Пользователь получил сообщение об ошибке, что нельзя редактировать свой профиль
Проверка выставления ошибок	Открыть приложение от лица аутентифицированного пользователя с правами на смену сценария у модуля. Сменить сценарий у модуля, который не отвечает	Модуль подсветился красным, при наведении будет написана причина ошибки
Множественные запросы	Открыть приложение от лица аутентифицированного пользователя с правами на смену сценария у модуля. Найти модуль на мини-карте, сделать двойной клик по модулю, не дожидаясь, пока придет запрос с сервера сделать двойной клик еще раз	На сервер отправится только один запрос на смену сценария, элемент будет неактивен до тех пор, пока не придет ответ с сервера
Сценарии этажей	Открыть приложение от лица аутентифицированного пользователя с правами на смену сценария у модуля. Открыть в выпадающем списке главной страницы здание, переключить сценарий, сменить сценарий у одного из модулей выбранного этажа на выкл, а один на вкл, сменить сценарий у всего этажа на вкл	Все модули этажа переключатся в состояние вкл