Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 - Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____ 2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Реализация комплексной системы управления процессами заселения в общежития АмГУ

Исполнитель
Студент группы 3105-ом1 _____ М.Е. Чирич
(подпись, дата)

Руководитель
доцент, канд. физ.-мат. наук _____ В.В. Ерёмина
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____ И.Е. Ерёмин
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.Н. Адаменко
(подпись, дата)

Рецензент
доцент, канд. техн. наук _____ С. Г. Самохвалова
(подпись, дата)

Благовещенск, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____ 2025 г.

З А Д А Н И Е

К магистерской диссертации студента группы 3105-ом1

Чирич Максим Евгеньевич

1. Тема магистерской диссертации: Реализация комплексной системы управления процессами заселения в об-щежития АмГУ

(Утверждено приказом от 06.03.2025 № 609-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.25

3. Исходные данные к магистерской диссертации: интернет-ресурсы, статьи, нормативные документы, техническое задание

4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): Компьютерные стратегии, проектирование алгоритма решения задачи, разработка модификации, применения современных методов разработки

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материла и т.п.):

6. Рецензент магистерской диссертации:

7. Дата выдачи задания 15.01.2025

Руководитель выпускной квалификационной работы:

В.В. Ерёмина, доцент, канд. физ.-мат. наук
(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 85 страницы, 46 рисунков, 30 источников

ВЕБ-ПРИЛОЖЕНИЕ, ПРОЕКТИРОВАНИЕ, ПРЕДМЕТНАЯ ОБЛАСТЬ, RUBY, RUBY ON RAILS, JAVASCRIPT, MVC, РАЗРАБОТКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, КЛИЕНТ-СЕРВЕР

Объект исследования – процесс заселения студентов в общежития Амурского государственного университета (АмГУ), учитывающий различные параметры и условия.

Цель работы заключается в разработке комплексной системы управления процессами заселения студентов в общежития АмГУ. Основными задачами являются оптимизация и автоматизация процесса заселения, повышение качества предоставляемых услуг, снижение административной нагрузки на сотрудников, а также обеспечение комфорта студентов в период заселения и в дальнейшем обучении.

Разработка включала в себя проектирование архитектуры системы, реализацию пользовательских интерфейсов, настройку интеграции с внутренними системами университета.

Разработанная система значительно повышает эффективность использования ресурсов университета и уровень удовлетворённости студентов. Она интегрирована в личный кабинет на сайте АмГУ и является первым этапом реализации системы «Общежитие». Взаимодействие с 1С:Университет осуществляется через специально разработанные web-сервисы, обеспечивающие синхронизацию данных и выполнение бизнес-логики. В основе системы – современные технологии веб-разработки: Ruby on Rails для серверной части и Vue.js для клиентской, что обеспечивает масштабируемость, модульность и удобство интерфейса.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных условиях одним из приоритетных направлений развития высших учебных заведений является цифровизация и автоматизация внутренних процессов. Это не только способствует повышению эффективности управления, но и напрямую влияет на качество образовательной среды и удовлетворённость студентов. Вопрос автоматизации отдельных административных процедур, таких как процесс заселения студентов в общежития, становится особенно актуальным в условиях роста числа обучающихся, усложнения регламентов и необходимости оперативной обработки больших объёмов данных.

Амурский государственный университет (АмГУ), как и многие другие вузы, стремится к созданию единого цифрового пространства, охватывающего все ключевые направления деятельности. Одной из задач в этом направлении является автоматизация процессов, связанных с размещением студентов в общежитиях, начиная от подачи заявок и учёта категории студентов до распределения мест и взаимодействия с другими информационными системами.

Целью настоящей диссертационной работы является разработка и внедрение информационной системы, автоматизирующей процесс заселения студентов в общежития АмГУ, с последующей интеграцией в существующую цифровую инфраструктуру вуза. Система должна обеспечивать прозрачность, удобство и надёжность всех этапов заселения, а также снижать нагрузку на сотрудников общежитий и административный персонал.

В ходе выполнения работы будут проанализированы существующие решения в сфере цифровизации вузов, выявлены потребности студентов и сотрудников, спроектирована архитектура информационной системы, реализован прототип и проведено его тестирование в учебной среде.

1 ОПИСАНИЕ ПРОЦЕССА ЗАСЕЛЕНИЯ СТУДЕНТОВ В ОБЩЕЖИТИЯ ФГБОУ ВО «АМГУ»

На текущий момент фиксируется большой рост количества поступающих в университеты, в том числе и в Амурский Государственный Университет при чем формат подачи заявлений абитуриентами осуществлялся преимущественно онлайн. Данная тенденция поддерживается общим технологическим развитием вузам по всей стране. Каждый вуз стремится в высокому уровне цифровизации, что в свою очередь не может не повлиять на другие процессы внутри университета.

Тенденция общей цифровизации коснулась сферу заселения студентов в общежития вуза. Она приобрела техническую реализацию в основе которой лежит разработка – модуль «Общежитие».

Основные принципы, поддерживаемые данной автоматизированной системой, наследуются от устаревшего ручного метода заселения, но имеет ряд технологичных изменений:

– электронная подача заявления;

– интерактивный процесс заселения студентов по их заявлениям;

– просмотра актуальных заявок на проживание;

– полный просмотр и анализ закреплённых мест в общежитии;

– системный расчёт и учет характеристик влияющих на рассмотрение заявлений студентов;

– отслеживание статуса заявления;

– одобрение или отклонение заявления с описанием причины.

1.1 Цели и задачи комиссии по заселению

Комиссия по заселению представляет собой специализированную организационную структуру, отвечающую за распределение студентов по местам в общежитиях вуза. Её деятельность направлена на планирование, координацию и контроль всех этапов процедуры заселения, при строгом соблюдении

установленных правил и норм, регулирующих проживание студентов. Основная цель комиссии – обеспечить эффективное и справедливое распределение койко-мест с учётом имеющихся ресурсов и актуальных потребностей студенческого сообщества.

Комиссия стремится создать комфортные, безопасные и благоприятные условия проживания, способствующие успешному обучению студентов. В её задачи входит разработка и согласование критериев приоритетности заселения, принимая во внимание как социальные, так и академические факторы. Также структура занимается организацией самого процесса заселения: подготовкой необходимых документов, распределением мест и формированием списка проживающих.

Важным направлением является своевременное информирование студентов о порядке и сроках подачи заявлений. Комиссия ведёт учёт свободных мест, регулярно обновляя базу данных жильцов общежитий, а также занимается урегулированием спорных ситуаций и конфликтов, возникающих в процессе размещения.

Работа комиссии по заселению студентов в общежития вуза заключается в проверке соблюдений правил подачи и верификации заявления, а также в эффективном распределении мест, обеспечении комфортных и безопасных условий проживания, оказании поддержки студентам и сотрудничестве с другими структурными подразделениями университета.

Ключевая задача комиссии – оптимальное распределение мест в общежитиях. Формируя списки заселения, комиссия строго соблюдает приоритеты (первокурсники, дети-сироты, инвалиды, иностранные студенты), а также учитывает пожелания студентов, их факультеты и курсы при расселении, что позволяет равномерно загружать корпуса и существенно снижает вероятность конфликтных ситуаций.

1.2 Организационная структура комиссии по заселению

Ключевые управленческие решения в процессе заселения студентов в общежития принимает специальная комиссия – «Комиссия по заселению».

Для более глубокого понимания её функций и распределения ролей необходимо рассмотреть внутреннюю организационную структуру этого органа. Подробная схема состава и взаимодействия участников комиссии представлена на рисунке 1.

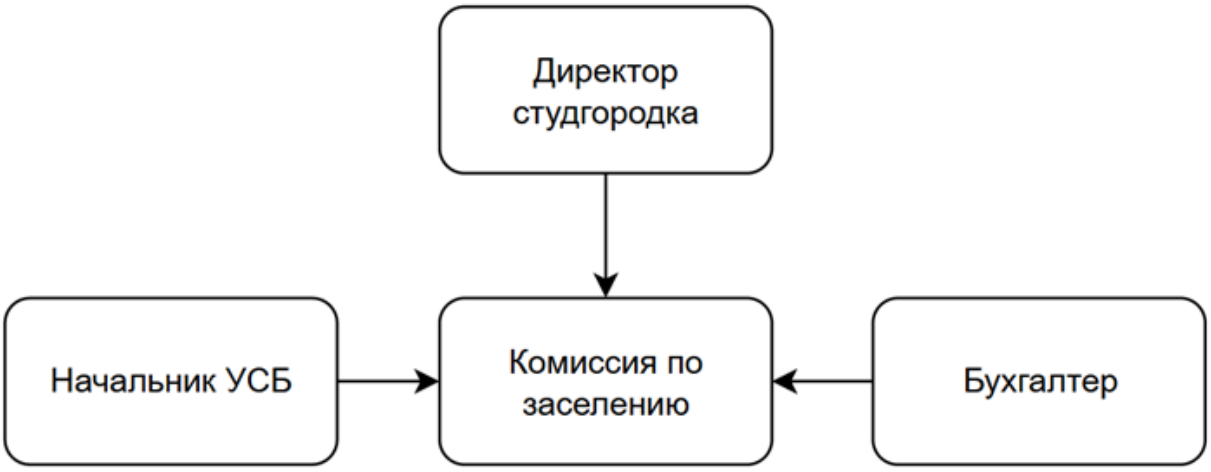

Рисунок 1 – Организационная структура комиссии

Основными участниками комиссии по заселению студентов в общежития являются несколько ключевых должностных лиц, каждое из которых выполняет строго определённую функцию в рамках своей компетенции.

Главный бухгалтер отвечает за финансовое сопровождение процесса заселения. Он контролирует учёт оплаты за проживание, проверяет и подписывает договоры, а также присваивает уникальные номера документам, обеспечивая унифицированную систему расчёта за предоставленное жильё.

Директор студенческого городка выполняет одну из центральных ролей в процессе распределения студентов по местам. Он несёт ответственность за общее управление и эксплуатацию общежитий, организует ремонтно-строительные работы и следит за их бесперебойной работой. Кроме того, он координирует распределение мест с учётом факультетской принадлежности, социальных факторов, а также особых условий заселения иностранных студентов. Данный участник фактически совмещает в себе задачи нескольких других членов комиссии, представляя интересы студентов и контролируя весь процесс расселения.

Директор управления собственной безопасности (УСБ) курирует вопросы, связанные с безопасностью проживающих, занимается оформлением сопутствующей документации и координирует взаимодействие с учебными подразделениями. Он обеспечивает организационную поддержку, направленную на защиту студентов в рамках процедуры заселения.

Комиссия по заселению в общежитие играет важную роль в процессе принятия и размещения студентов. Ее основная задача заключается в организации и проведении процедуры заселения, обеспечивая справедливость и эффективность этого процесса. Наряду с этим рассматривает различные факторы, такие как доступность мест, приоритеты, установленные регламентом или политикой общежития, и особые обстоятельства, как правило привилегий заявителей. Данная структура описывает реальные данные о содержании комиссии, ниже описывается существующие ПО, в котором состав людей, взаимодействующих с системой, отличен от текущего, что по своей сути уже является предметом для исправлений и доработок.

1.3 Обзор существующего метода решения рассматриваемой задачи

Функциональное отображение механизмов деятельности взаимосвязанных структур в ранее разработанной системе «Общежитие» далее будут описаны с помощью структурно-функциональной модели стандарта IDEF0.

Диаграмма, изображённая на рисунке 2, представляет внешний уровень обработки данных в системе заселения и наглядно демонстрирует, как осуществляется взаимодействие между пользователями и системой. В частности, на ней отображены ключевые участники процесса – это студенты различных курсов, подающие заявления на заселение, и члены комиссии, ответственные за их рассмотрение, верификацию, распределение мест и принятие решений.

Среди входных данных указаны сведения, вводимые студентами при подаче заявлений: персональная информация, предпочтения по выбору общежития и комнаты, а также сопутствующие документы. Данные поступают в систему и становятся доступными для обработки комиссией. На выходе формируются результаты рассмотрения – это либо одобрение, либо отказ, а также

распределение по конкретным местам в общежитии. Комиссия по заселению играет роль центрального органа, обеспечивающего контроль над корректностью данных, соблюдением критериев распределения, учётом социальной категории, курса обучения, академических и иных оснований. Диаграмма акцентирует внимание на информационных потоках между различными участниками, помогая визуально понять, как происходит обработка заявлений на уровне системы и как обеспечивается координация между ее элементами.

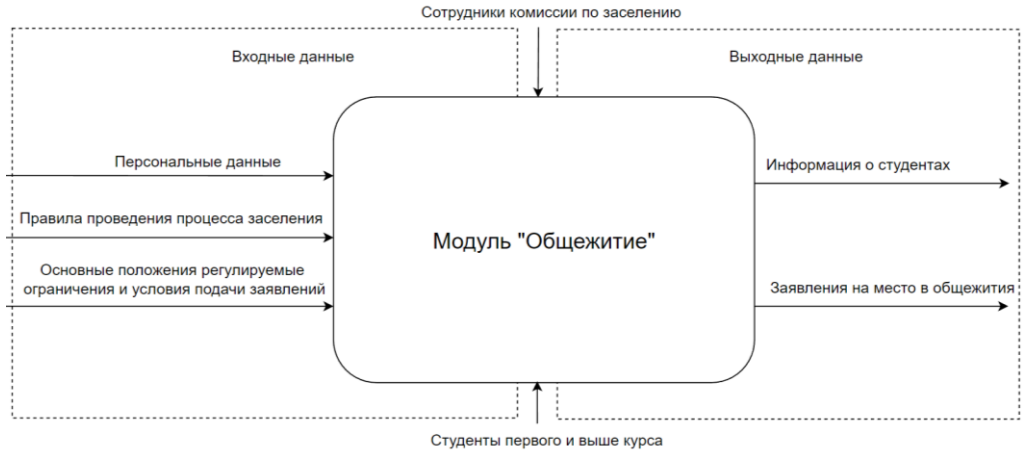

Рисунок 2 – Структурно-функциональная модель модуля «Общежитие»

Декомпозиция системы, изображенная на рисунке 3, описывает внутренние процессы модуля, которые связаны с его основной функциональностью. Она также представляет сущности, которые взаимодействуют с внешними входными данными.

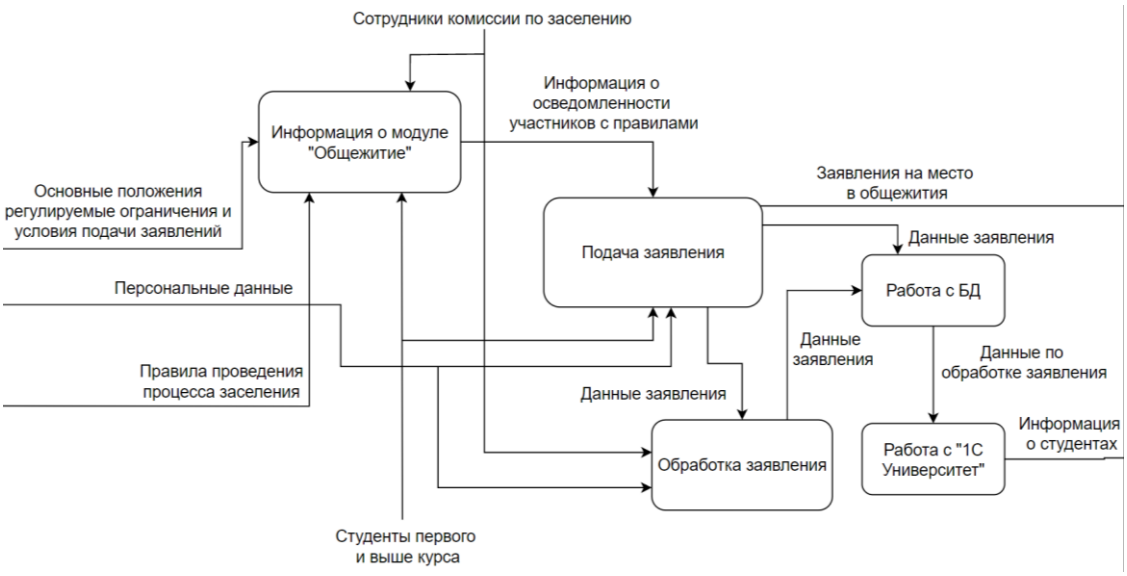

Рисунок 3 – Декомпозиция модуля «Общежитие»

На ряду с этим необходимо рассмотреть основные описанные сущности, «Подача заявления» и «Обработка заявления», для полного раскрытия сути автоматизации используется декомпозиция, представленная на рисунках 4 и 5.

Рисунок 4 – Декомпозиция модуля «Подача заявления»

Декомпозиция модуля «Подача заявления», описывает какие манипуляции совершаются студентами, а также какую информацию он получает от системы.

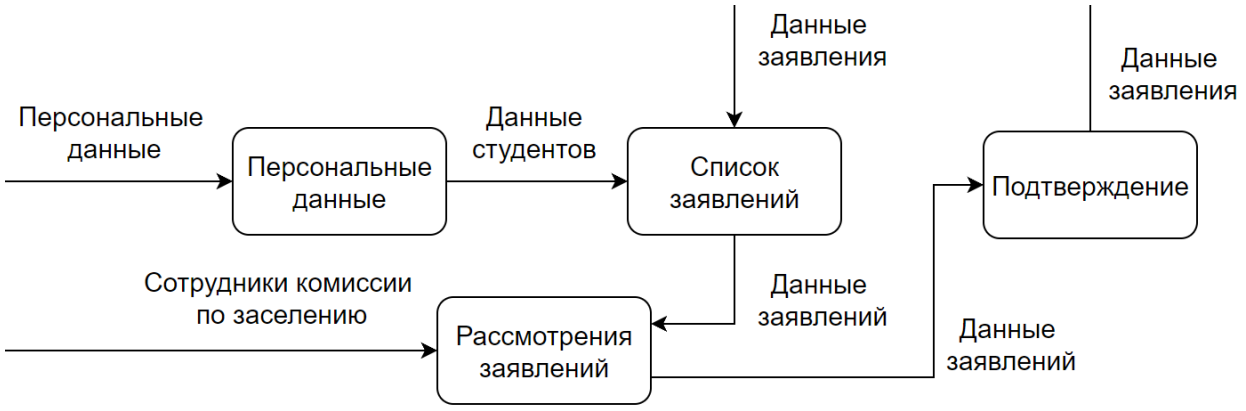

Рисунок 5 – Декомпозиция модуля «Обработка заявления»

Декомпозиция модуля «Обработка заявления» раскрывает структуру и функциональные возможности, предоставляемые сотрудникам комиссии в рамках работы с поданными заявлениями. Кроме того, в описании уделяется внимание входным и выходным данным, обрабатываемым данным модулем.

В рамках анализа текущей системы были выделены основные категории пользователей (актеров), взаимодействующих с модулем: студент, бухгалтер, директор студенческого городка, начальник Управления студенческой безопасности (УСБ), комендант общежития, а также заместитель декана по воспитательной работе.

Студенту предоставлен доступ к ряду функций, среди которых: просмотр актуальных правил подачи заявлений, проверка наличия свободных мест в общежитиях, относящихся к его факультету, ознакомление с личным рейтингом академической успеваемости, а также возможность подачи и последующего редактирования собственного заявления. Все перечисленные действия отражены в виде вариантов использования (use case) на соответствующей диаграмме. Между подачей заявления и его включением в общий список существует логическая связь, указывающая на автоматическую передачу данных в систему обработки.

Сотрудники комиссии – включая главного бухгалтера, директора студгородка, начальника управления собственной безопасности и коменданта общежития – наделены правами доступа к заявлениям студентов, относящимся к конкретному факультету. Они имеют возможность просматривать и проверять поданные заявления, одобрять их, а при необходимости – получать доступ к содержащейся в них персональной информации.

Особую роль в системе занимает заместитель декана по воспитательной работе. Его функционал во многом совпадает с полномочиями других членов комиссии, однако с одним важным отличием – он не участвует в распределении студентов по местам, выбранным в заявлении.

Логика взаимодействия всех указанных пользователей с системой отображена на диаграмме, приведённой на рисунке 6.

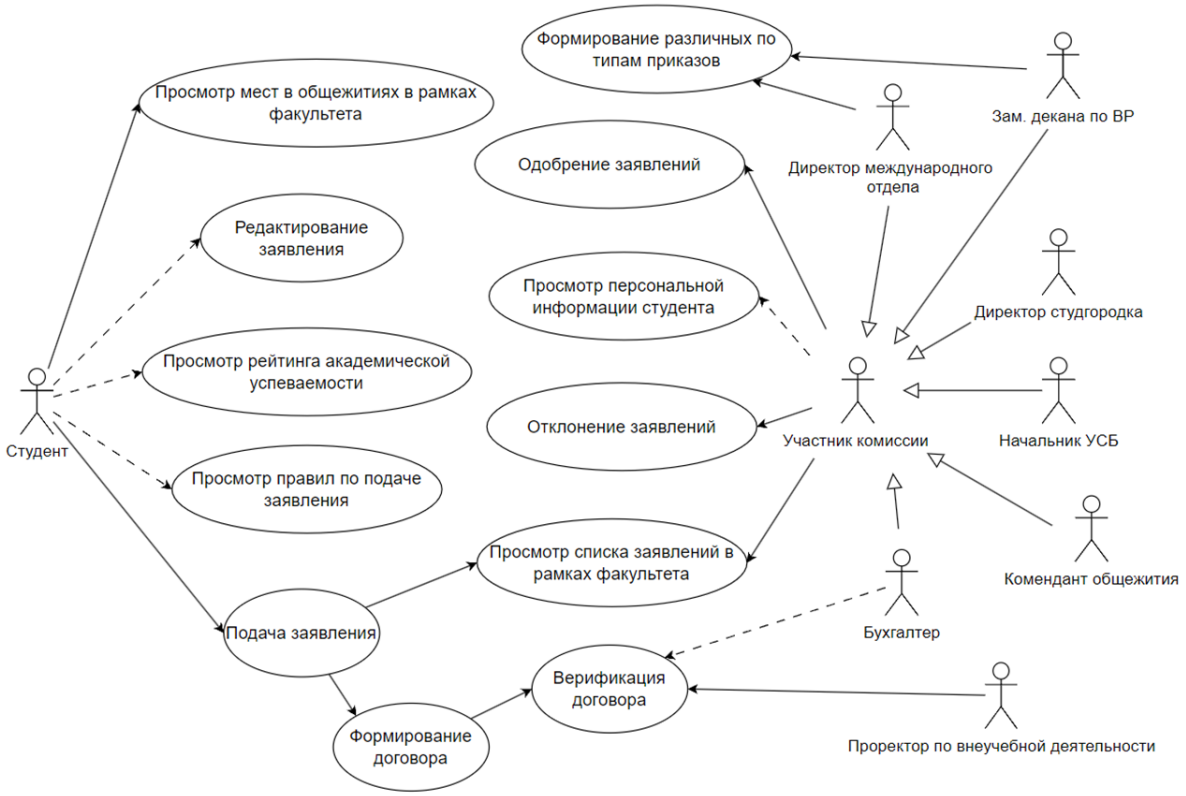

Рисунок 6 – Диаграмма вариантов использования

На следующем этапе проектирования системы представлены диаграммы последовательности, позволяющие наглядно отобразить процессы взаимодействия между ключевыми объектами и компонентами приложения.

Одной из важнейших является диаграмма, иллюстрирующая шаг за шагом процесс подачи студентом заявления на предоставление места в общежитии. В ней отражены не только формирование и передача данных, но и этапы проверки корректности введённой информации и получение ответа от системы. Детальное визуальное представление данного процесса показано на рисунке 7.

Процесс начинается с того, что студент получает доступ к нормативным документам и правилам подачи заявлений, после чего знакомится с информацией о наличии свободных мест в общежитиях факультета. Изучив данные, он выбирает подходящие варианты и формирует заявление. Затем оно передаётся

на сервер.

На стороне сервера происходит автоматическая проверка (валидация) полученного заявления. Если проверка пройдена успешно, пользователю возвращается сформированное заявление, сохранённое в базе данных, с возможностью его дальнейшего редактирования при необходимости. Дополнительно заявление включается в общий список поданных заявлений, доступных для обработки соответствующими сотрудниками.

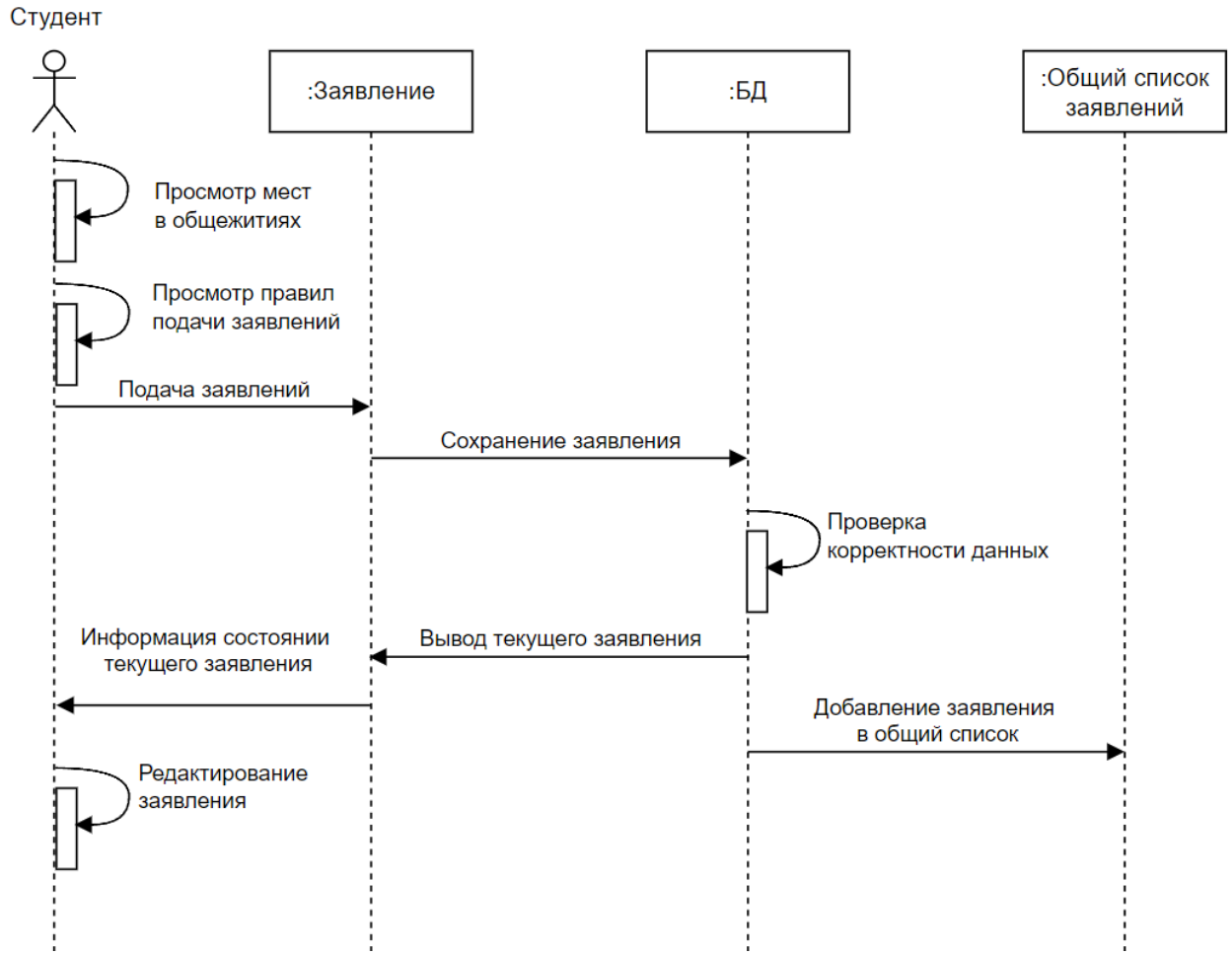

Рисунок 7 – Диаграмма последовательности студента

Вторая диаграмма последовательности, представленная на рисунке 8, иллюстрирует процесс получения списка заявлений сотрудниками комиссии и последующую их верификацию. Этап является важной частью системы обработки заявлений, обеспечивая контроль за корректностью и обоснованностью поданных данных.

После того как заявления собраны, система распределяет их между членами комиссии. Каждый представитель получает перечень заявлений, подлежащих рассмотрению. Далее начинается этап анализа: представители комиссии просматривают предоставленные данные, сверяют их с установленными требованиями и проводят все необходимые проверки.

На основании полученной информации комиссия принимает решение – удовлетворить заявление или отклонить его. Результаты рассмотрения фиксируются в общей базе, где отражается текущее состояние каждого заявления.

Завершённые и одобренные заявления попадают в общий список, откуда они передаются к следующему этапу – распределению. За часть процесса отвечает специально назначенный представитель комиссии. Его задача – перенести данные одобренных заявлений в базу данных, а также оформить соответствующие документы и закрепить их в информационной системе 1С Университет.

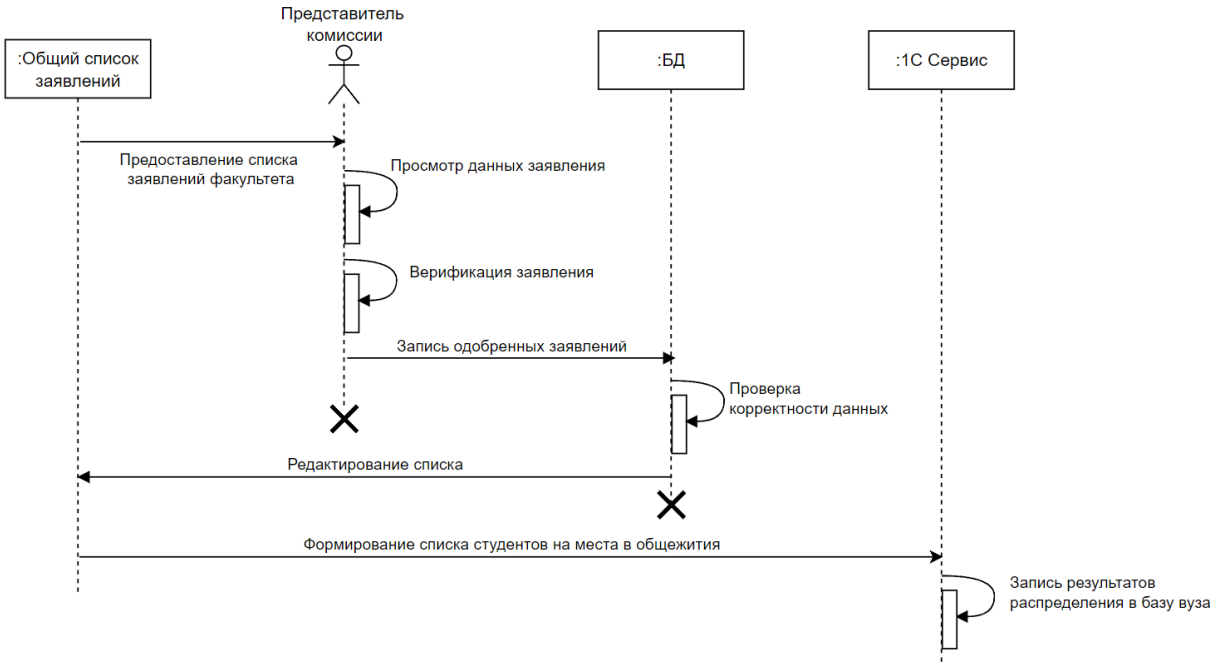

Рисунок 8 – Диаграмма последовательности верификации заявлений

На текущий момент система имеет функциональный интерфейс, представленный на рисунке 9 и рисунке 10.

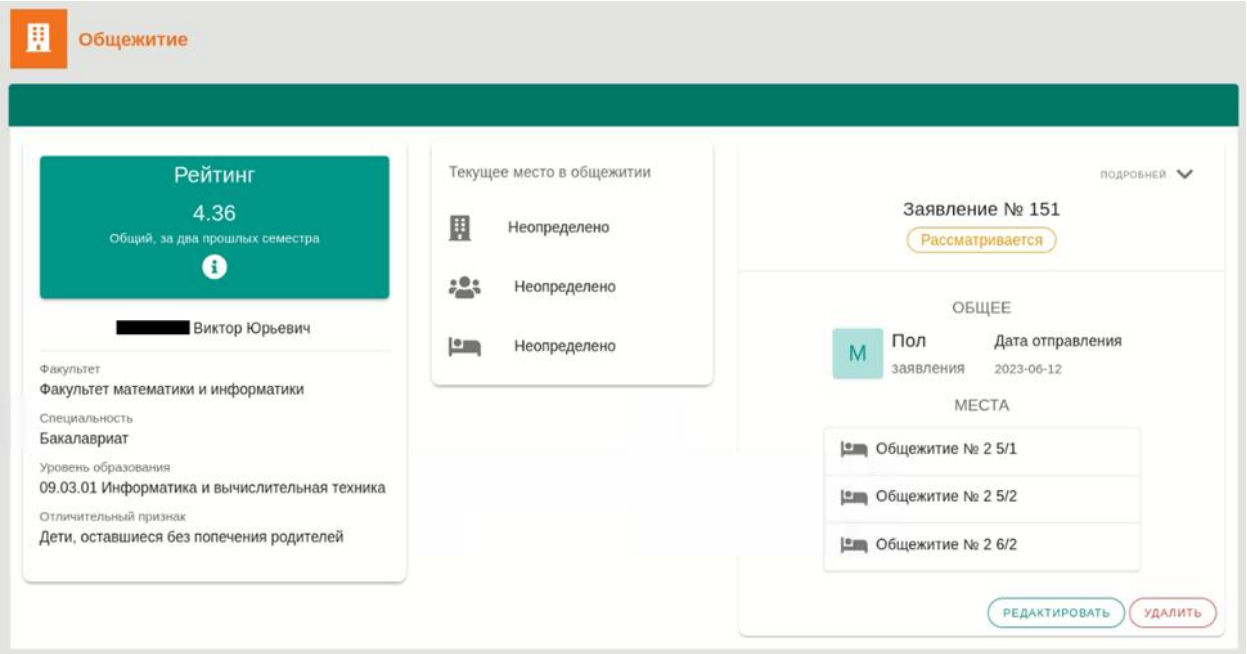

Рисунок 9 – Вид главного окна студента

Интерфейс главного окна студента представляет из себя набор базовых функций, направленных на подачу заявлений и отслеживание его статусов. На ряду с этим имеется отображение статусов комиссии, текущего места жительства и основной информации заселяющегося.

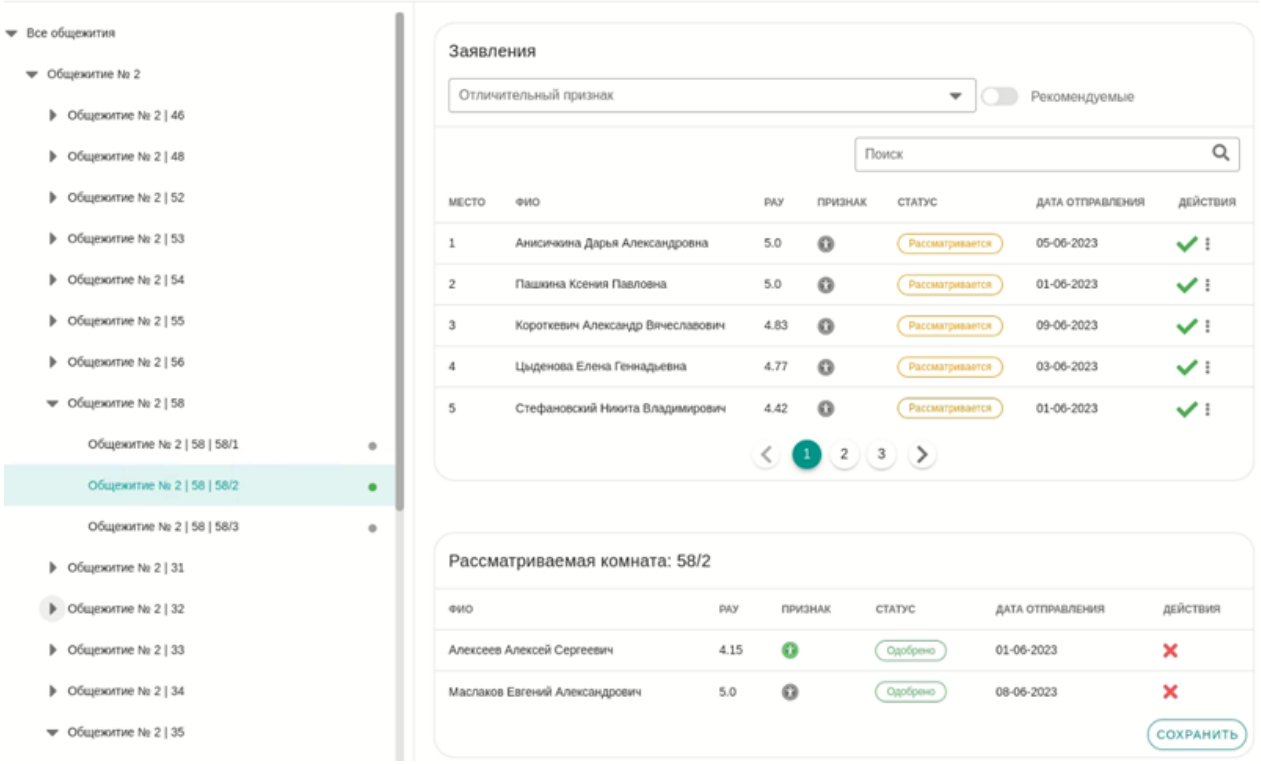

Рисунок 10 – Распределение студентов по местам

Достаточно хорошим функционалом обладает интерфейс участника комиссии, отвечающего за закрепление студентов за места в общежитиях. Имеется отображение всем комнат, заявлений студентов, а также интерактивное распределение с показом различных информативных меток.

Из общего анализа можно выявить ряд ключевых особенностей, подлежащие улучшение или замене для нормального функционирования на текущий момент:

– сокращение числа людей в комиссии. Объемный функционал распределяется не равномерно и многие решения остаются не прозрачными или неуместными. Необходимо выделить одного участника, который мог видеть полную и развернутую информацию по заполненности и структуру в целом общежитий;

– интерактивный интерфейс подразумевает под собой только заселение и выселение. Необходимо расширение функционала в виде добавлений переселения, а также статистики и определенных расширенных метрик;

1.4 Основные положения регулирующие процессы разрабатываемого модуля

Положение о структурных подразделениях образовательной организации представляет собой локальный нормативный документ, регулирующий деятельность внутренних служб вуза. Оно устанавливает порядок взаимодействия между подразделениями, распределение функций и ответственность каждого участника в рамках различных процессов, в том числе заселения студентов в общежития.

Одним из таких положений является документ о жилищно-бытовой комиссии студенческого городка Амурского государственного университета (АмГУ) – ПКО СМК 22-2013. Он разработан на основе норм жилищного законодательства, правил внутреннего распорядка в общежитиях АмГУ, положений о студенческом совете общежитий и студгородке. Положение определяет цели, задачи, состав и функции комиссии, обеспечивающей порядок и справедливость в процессе распределения мест.

Основная цель комиссии – контроль за соблюдением правил проживания в общежитиях, а также организация процесса заселения с учетом нормативов и интересов всех участников. Возглавляет комиссию проректор по экономике и перспективному развитию. В состав могут входить: главный бухгалтер, директор и заместители директора студгородка, заместители деканов по воспитательной работе всех факультетов, представители отделов безопасности, юридической службы, международных проектов и студенческого самоуправления. При этом состав комиссии может корректироваться в зависимости от конкретных задач, что делает структуру гибкой и адаптируемой.

Важную роль играют Правила внутреннего распорядка в общежитиях (ПР СМК 7.1.01-2013), которые действуют на всей территории студгородка АмГУ. Они описывают порядок заселения, основания для выселения, условия проживания и оплаты. Заселение проводится с соблюдением санитарных норм – не менее 6 кв.м. на одного человека. Распределение мест осуществляется студгородком, а само заселение – деканатами факультетов на основании заявлений студентов. Временный пропуск в общежитие выдается службой безопасности на срок до 10 дней до получения регистрации и постоянного пропуска. Обязательным является заключение договора найма и предоставление медицинской справки.

Дополнительно в университете действует положение о дисциплинарных взысканиях, регулирующее основания и порядок привлечения студентов к ответственности за нарушения устава, правил внутреннего распорядка и иных локальных актов. Документ предусматривает защиту прав обучающихся, особенно с ограниченными возможностями здоровья, а также устанавливает запрет на применение наказаний в период болезни, отпуска или каникул. Каждое нарушение может повлечь за собой лишь одну меру дисциплинарного взыскания.

Таким образом, нормативная база АмГУ обеспечивает чёткую организацию процесса проживания студентов, способствует формированию безопасной и справедливой среды в общежитиях, а также гарантирует соблюдение

прав и обязанностей всех участников учебного процесса.

1.5 Анализ использования программно-технических средств

В Амурском государственном университете (АмГУ) внедрена современная информационная система «1С: Университет», которая является универсальным инструментом для автоматизации и координации основных направлений деятельности вуза. Программное решение охватывает широкий спектр задач, начиная от приема абитуриентов и заканчивая сопровождением выпускников после окончания обучения. В частности, с помощью системы осуществляется контроль над учебным процессом, ведется учет посещаемости и успеваемости студентов, производится расчет и начисление платы за обучение, а также планируется и распределяется учебная нагрузка преподавателей.

Отдельное внимание уделено сопровождению студентов на всех этапах их обучения. Система позволяет отслеживать академический прогресс, формировать расписания, назначать экзамены и зачеты, а также организовывать мероприятия, связанные с трудоустройством выпускников. Таким образом, «1С: Университет» выступает как основа цифровой образовательной среды вуза.

В дополнение к этой системе в АмГУ функционирует личный кабинет обучающегося и преподавателя – онлайн-платформа, обеспечивающая индивидуальный доступ пользователей к учебной информации и возможностям взаимодействия с университетом. Через личный кабинет студенты могут следить за своими академическими результатами, графиком занятий, посещаемостью, выполнять и загружать задания, а также участвовать в научной деятельности. Кроме того, в личном кабинете реализована возможность обратной связи: студенты и преподаватели могут обмениваться сообщениями, получать уведомления и быть в курсе последних новостей университета.

Платформа личного кабинета поддерживает дистанционный формат обучения, что особенно важно в условиях удалённого или смешанного образовательного процесса. В ней предусмотрена интеграция с функционалом «1С: Университет», благодаря чему пользователи получают доступ к широкому

спектру цифровых сервисов. Например, студент может подать заявление, просматривать сведения о начислениях и платежах, запрашивать справки и документы. В то же время, административные права доступа к управлению данными сохраняются за уполномоченными структурами: деканатами, кафедрами и администрацией университета. Такой подход обеспечивает высокую степень безопасности, защищает персональные данные и исключает несанкционированное вмешательство в образовательные процессы.

Одной из значимых возможностей цифровой инфраструктуры вуза является интеграция с внешними сервисами, такими как API «Контур-Сайн». Данное решение позволяет использовать механизм неквалифицированной электронной подписи (НЭП) при оформлении договоров, например, на проживание в общежитии. Благодаря этой интеграции студенты могут подписывать необходимые документы в электронном виде, не посещая университет лично. Это особенно удобно для иногородних и иностранных студентов, а также позволяет ускорить обработку документов и минимизировать бумажный документооборот. После подписания договор автоматически передается и сохраняется в системе «1С: Университет», что обеспечивает прозрачность, юридическую значимость и точность хранения данных.

1.6 Анализ требований к разрабатываемой системе

Разрабатываема система охватывает несколько функциональных процессов, связанных с общежитиями, которые имеют особенность претерпевать значительные технические и концептуальные изменения. Данное поведение обусловлено рядом прикладных причин, включающих в себя новые потребности в работе заселения и его этапов проведения.

В системе «Общежитие» для студентов университета возникла необходимость в различных типах заявлений, поскольку изменяется парадигма заселения, а именно заселение в комнату в общежитии производится один раз и на весь период обучения. Это требует создания разных форм заявлений для учета всех возможных ситуаций и потребностей студентов, таких как переезд в другую комнату, временное освобождение комнаты, замена жильцов, ремонт и

другие вопросы, связанные с проживанием в общежитии.

В процессе подачи заявлений на заселение в общежития университета выделяют три основных типа заявлений, каждый из которых имеет свои уникальные особенности и требования. Эти типы можно классифицировать по их функциональному назначению:

– заявление на заселение. Основной целью назначения данного типа заявления является обеспечение первичного размещения студента в общежитии. Требует предоставления информации о студенте, такой как личные данные, учебная группа, факультет и курс. Включает указание более предпочтительной для самого студента комнаты. Подразумевает рассмотрение комиссией по распределению мест в общежитиях, которая оценивает приоритеты и возможности предоставления жилья;

– заявление на выселение. Этот тип заявления предназначен для официального уведомления администрации общежития о намерении студента освободить занимаемое жилье. Содержит причину выселения (например, окончание учебы, перевод в другой университет, личные обстоятельства). Включает дату планируемого выселения. Может требовать подтверждения отсутствия задолженностей по оплате за проживание и состояния жилого помещения. После подачи заявления проводится финальная проверка комнаты и расчет окончательных платежей;

– заявление на переселение. Данный тип заявления используется студентами, которые уже проживают в общежитии, но желают сменить комнату или здание общежития. Требует обоснования причины переселения (например, неудовлетворительные условия проживания, конфликт с соседями, желание смены обстановки). Включает информацию о текущем месте проживания и желаемом новом месте. Подразумевает рассмотрение комиссией по заселению, которая оценивает возможность удовлетворения запроса с учетом наличия свободных мест и иных факторов.

Вместе с расширением типов заявлений вводится изменение в системе проживания. В настоящий момент рассмотрено и принято правило, согласно

которому студент, желающий проживать в общежитии, получает место на весь срок обучения.

– требуется разработать и внедрить обновленный алгоритм рассмотрения заявлений, который обеспечит более точное и оперативное принятие решений по заселению. В этом контексте особое внимание уделяется использованию электронных подписей членов комиссии для повышения эффективности данного этапа;

– необходимо интегрировать систему общежития с базой данных 1С для автоматической проверки нахождения студентов в приказе. Это обеспечит более надежное и быстрое подтверждение данных и сэкономит время на административных процедурах;

– осуществить обновление интерфейса для студентов, что позволит им более эффективно отслеживать и контролировать движение своих заявлений, а также улучшит общую пользовательскую удовлетворенность.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

В условиях современного образовательного учреждения, где большое значение придается автоматизации и эффективности процессов, разработка и долгосрочная поддержка системы «Общежитие» является важной задачей. Далее рассматриваются ключевые аспекты проектирования, начиная с анализа требований и заканчивая архитектурными решениями для приложения и описания его функциональных модулей.

Особое внимание уделяется взаимодействию различных участников процесса и интеграции новых функциональных возможностей, направленных на улучшение обработки заявлений студентов.

2.1 Связь и взаимодействие действующих лиц системы

На основании существующей системы с учетом новых требований к её функциональности была модифицирована диаграмма прецедентов, описывающая взаимодействия новых пользователей и их возможностей (рисунок 11).

Главным новшеством стало участие в автоматизированном процессе заселения студентов международного отдела, который имеет свой уникальный процесс заселения. Он базируется на отметке нуждаемости студента или абитуриента в проживании в общежитии университета. Часто возникают трудности из-за языкового барьера, который мешает конструктивно выявлять у иностранного студента необходимую информацию. В связи с этим, данное нововведение имеет положительный характер. Студенты, имеющие такие отметки, автоматически попадают в распределение по местам, закрепленным за международным отделом. Директор международного отдела проводит расселение и формирует приказы на основании полученных данных.

Дополнительно, процесс был адаптирован для улучшения взаимодействия с другими административными единицами университета, что позволило повысить прозрачность и эффективность всей системы. Студенты получают

возможность формирования актуального и адаптивного договора на общежитие, который учитывает как нормативные требования, так и персональные данные, необходимые для заполнения. После формирования договора он проходит верификацию проректором и бухгалтером.

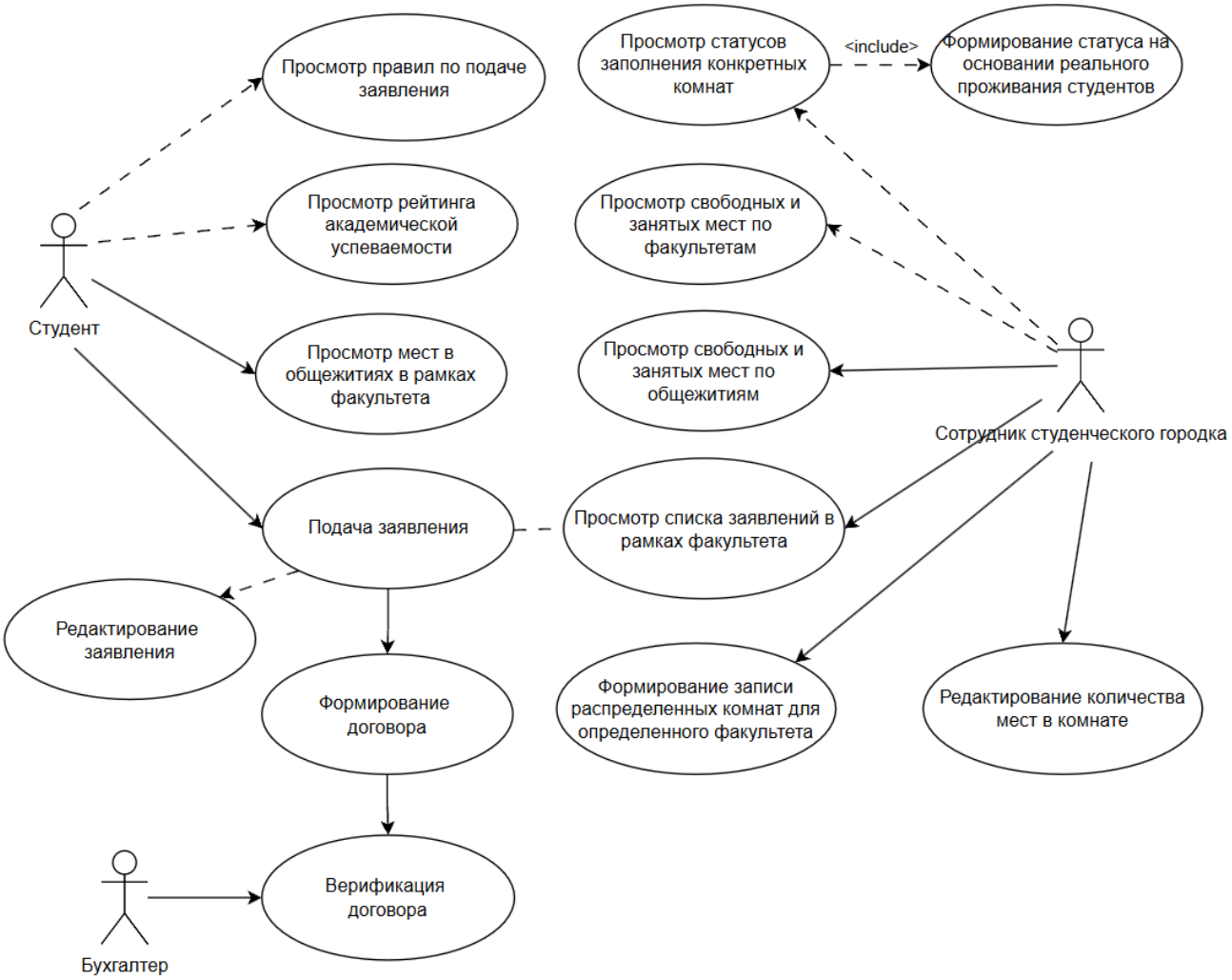

Рисунок 11 – Модифицированная диаграмма вариантов использования

Для студента также добавляется возможность формирования актуального и адаптивного договора на проживание в общежитии. Этот договор учитывает как нормативные требования, так и персональные данные студента, необходимые для заполнения. После формирования договора следует его верификация, осуществляемая проректором и бухгалтером. Введение данной функциональности обусловлено большой текучестью документов на про-

верку, что требует автоматизации процесса. Кроме того, существует возможность доработки договора, что позволяет сделать процесс более гибким и адаптивным к изменениям.

2.2 Описание функциональных решений для реализации системы

Процесс начинается с проверки типа действия, рисунок 12.

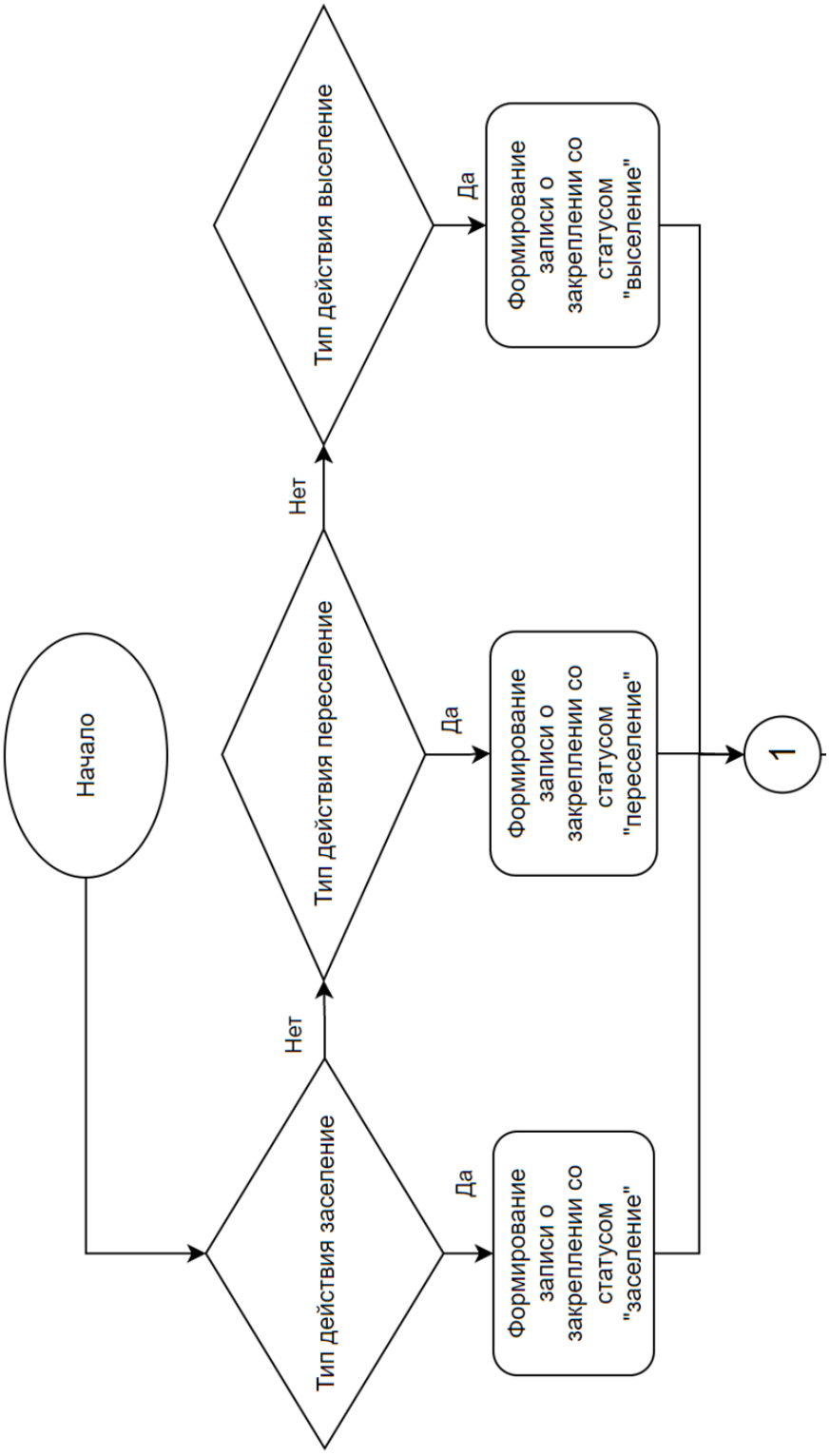

Рисунок 12 – Диаграмма алгоритма обработки заявления, первый этап

Первым проверяется, является ли действие выселением. Если это так, формируется запись о закреплении со статусом «выселение». В противном случае, если действие не является выселением, проверяется следующий тип действия – переселение. Если это переселение, то формируется соответствующая запись о закреплении со статусом «переселение». Если же действие не является ни выселением, ни переселением, проверяется, является ли оно заселением. Если да, то формируется запись о закреплении со статусом «заселение».

Таким образом, в зависимости от типа действия (выселение, переселение или заселение) система формирует соответствующую запись.

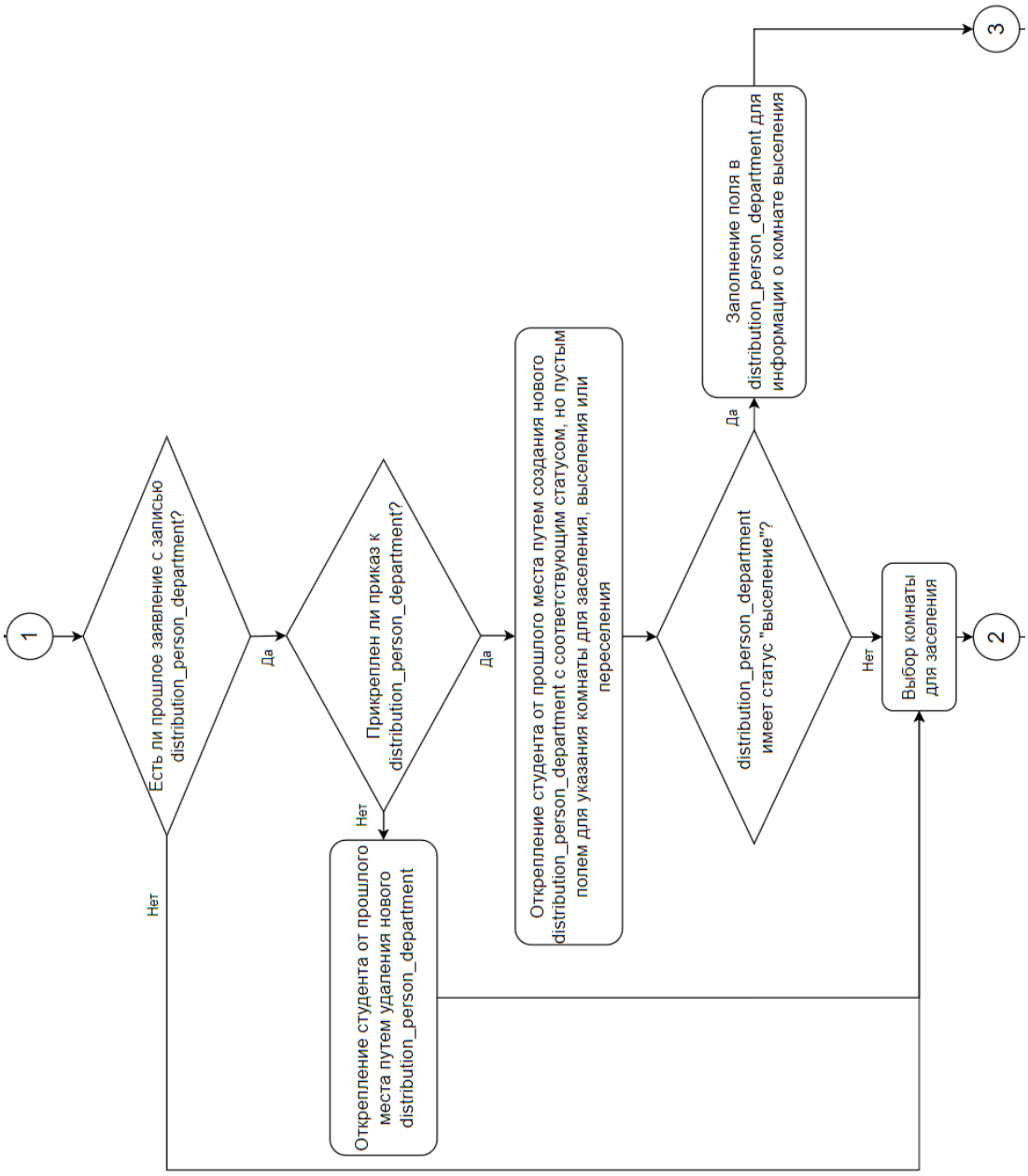

Рисунок 13 – Диаграмма алгоритма обработки заявления, второй этап

Далее на рисунке 13 описывается дальнейшие шаги в процессе управления заселением, переселением и выселением студентов после формирования записи о закреплении.

На первом этапе проверяется наличие заявления с записью в поле distribution_person_department, которое отвечает за хранение информации о текущем месте жительства студента, закрепленном за определенным местов в общежитии. Если заявление присутствует, проверяется наличие приказа, связанного с этим заявлением. Если приказ имеется, происходит открепление студента от предыдущего места путем удаления информации из distribution_person_department. Далее проверяется, связано ли освобожденное место с пустым статусом «заселение», «выселение» или «переселение». Если это так, осуществляется заполнение поля distribution_person_department новой информацией о комнате или месте выселения, и процесс переходит на следующий этап.

Если заявление отсутствует, проверяется статус выселения для текущей записи. Если статус выселения установлен, происходит открепление студента от текущего места в distribution_person_department. Если статус выселения не установлен, проверяется наличие свободной комнаты для заселения. В случае наличия свободной комнаты, процесс продолжается с её использованием. Данный фрагмент блок-схемы описывает логический алгоритм обработки заявлений, проверки статусов и управления местами в общежитии, обеспечивая последовательность и точность выполнения операций в системе. Поле distribution_person_department играет ключевую роль в хранении и управлении информацией о размещении студентов, что позволяет автоматизировать и упростить процессы заселения, переселения и выселения.

Продолжение блок-схемы, а конкретнее рисунок 14 описывает завершающие этапы процесса управления заселением, переселением и выселением студентов. После проверки наличия свободного места в комнате, если место есть, проверяется наличие нового заявления с записью в поле

distribution_person_department, которое хранит информацию о текущем месте жительства студента.

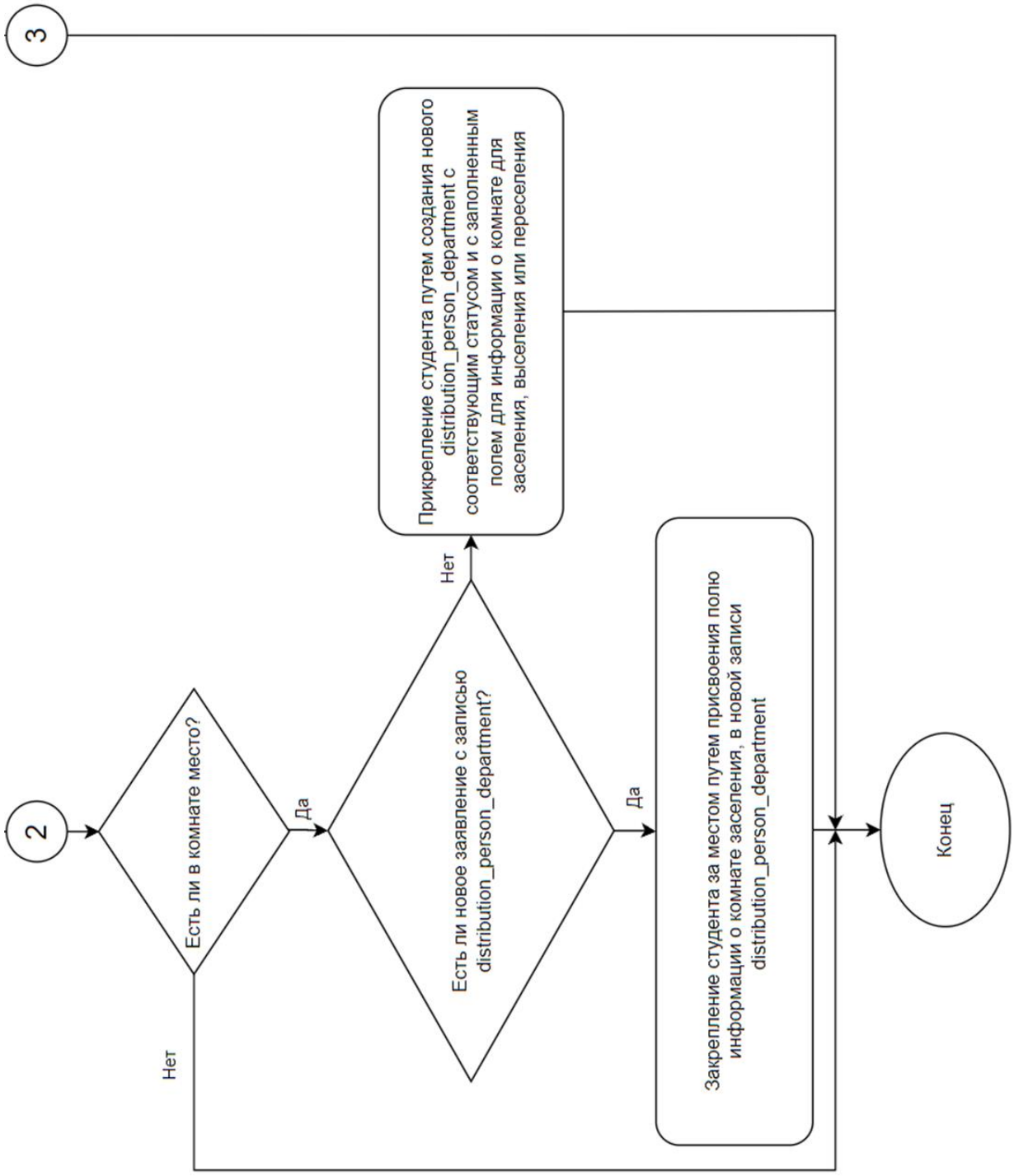

Рисунок 14 – Диаграмма алгоритма обработки заявления, третий этап

Если новое заявление присутствует, студент закрепляется за новым местом с заполнением информации о комнате в distribution_person_department. В случае отсутствия заявления, студенту создается новая запись в этом поле с

соответствующим статусом. Если свободного места в комнате нет, процесс возвращается к предыдущим этапам для дальнейшей проверки.

Общая блок-схема описывает процесс автоматизированного управления заселением, переселением и выселением студентов, включая проверку наличия мест и обработку заявлений. Поле distribution_person_department играет ключевую роль в хранении информации о текущем статусе размещения студентов. В результате, система обеспечивает точность, последовательность и автоматизацию операций по управлению местами в общежитии.

2.3 Описание функциональных модулей и их взаимодействие в разрабатываемой системе

В архитектуре системы заселения студентов в общежития университета применяется модульный подход, обеспечивающий гибкость, масштабируемость и упрощение управления функциональностью.

Подход основан на разделении функционала на отдельные модули, каждый из которых отвечает за определенные аспекты системы. Это позволяет легко добавлять новый функционал, модифицировать или удалять существующий без значительных изменений в других частях системы. Модульный подход также способствует улучшению качества обслуживания, снижению вероятности ошибок и повышению общей эффективности системы.

Система состоит из нескольких ключевых компонентов, изображенных на рисунке 15. Центральным узлом является Личный кабинет, обеспечивающий интерфейс взаимодействия пользователей с системой. Модули представляют собой специализированные компоненты, каждый из которых отвечает за определенный набор функций, таких как регистрация, управление заявками, распределение мест и формирование документов. База данных (1С Университет) служит централизованным хранилищем данных, обеспечивая надежное хранение и управление всеми необходимыми данными. Распределение задач позволяет системе эффективно обрабатывать большой объем информации и оперативно реагировать на изменения в требованиях и процессах.

Функциональные модули:

– модуль «Подача заявлений» предназначен для оформления заявок на предоставление мест в общежитии и управления сопутствующей информацией. Пользователь может отслеживать статус поданного заявления, редактировать персональные данные и прикреплённые документы. Взаимодействие реализуется через личный кабинет, который обращается к базе данных для сохранения, получения и обработки информации. Таким образом, обеспечивается полноценный цикл подачи и сопровождения заявлений в цифровом формате;

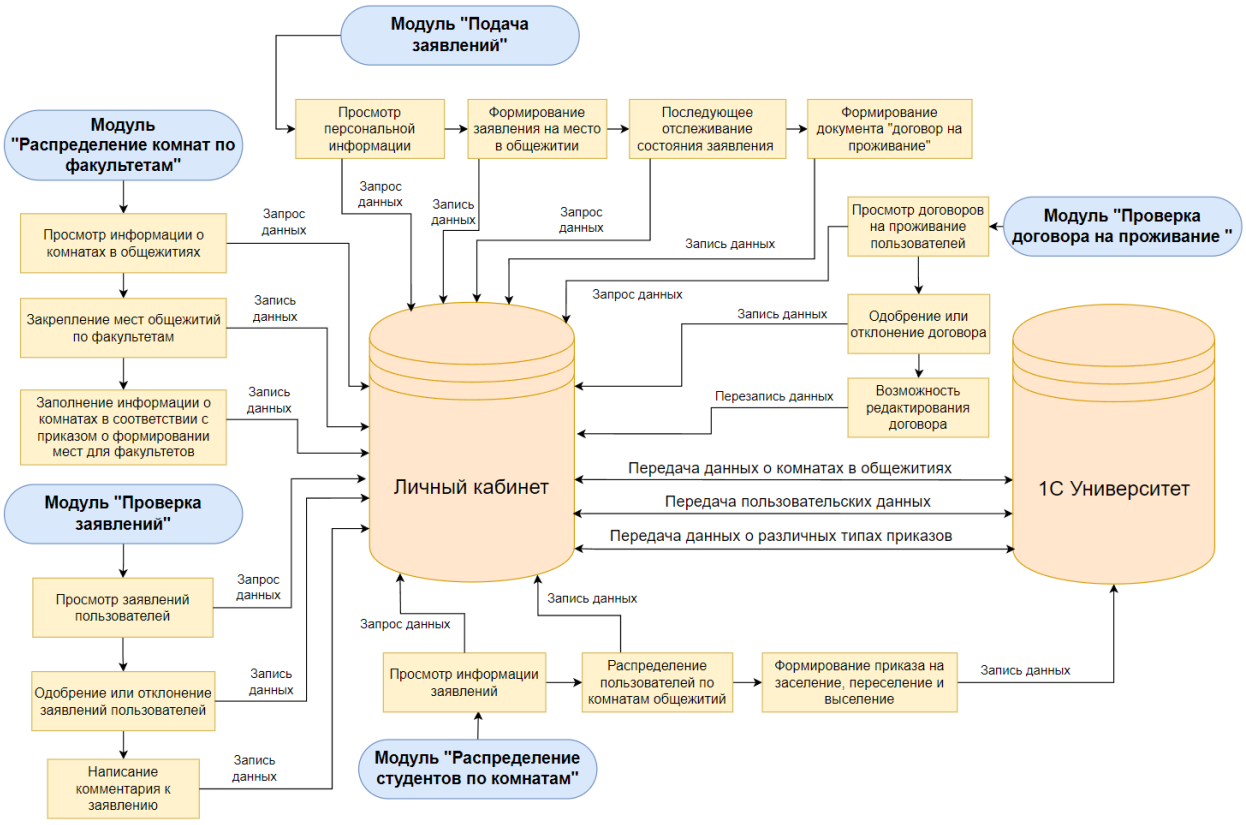

Рисунок 15 – Представление модульного подхода для системы «Общежитие»

– модуль «Распределение комнат по факультетам» позволяет закреплять места в общежитии за конкретными факультетами. Обеспечивает управление информацией о доступных комнатах. Выполняет подготовку приказов о распределении мест. Также реализует запрос и запись данных в личный кабинет и базу данных, что позволяет синхронизировать информацию о комнатах и приказах;

– модуль «Проверка заявлений» захватывает и выполняет ряд функций:

рассмотрение поданных заявлений; принятие решений об одобрении или отклонении заявлений; добавление комментариев к заявлениям; запрос данных из личного кабинета для анализа заявлений, запись решений обратно в систему;

– модуль «Распределение студентов по комнатам» реализует распределение студентов по комнатам на основе одобренных заявлений, а также управление приказами на заселение, переселение и выселение студентов. Запрос данных из личного кабинета и запись результатов распределения в базу данных;

– модуль «Проверка договора на проживание» имеет функционал по управлению договорами на проживание, одобрению или отклонению договоров. На ряду с эти определена возможность редактирования и обновления договоров. Запрос и запись данных из личного кабинета и в базу данных для обеспечения актуальности информации о договорах.

Взаимодействием между компонентами выступает Личный кабинет: он обращается к базе данных для получения необходимой информации для каждого модуля. После обработки информации модули передают результаты обратно в личный кабинет, который обновляет базу данных.

Для обеспечения корректной и эффективной работы вышеописанных компонентов системы, можно выделить несколько вспомогательных сервисов бизнес-логики, которые будут выполнять промежуточные вычисления и дополнительные операции для поддержки основной функциональности. Эти сервисы могут включать в себя:

– сервис расчета доступных мест. Этот сервис будет выполнять вычисления для определения, сколько мест доступно на каждом факультете и в каждом здании общежития в реальном времени. Он может учитывать текущую загрузку комнат, а также временные параметры, такие как даты начала и окончания учебных периодов. Также он может работать с данными о новых заявках, делая расчеты по ожидаемым изменениям в размещении студентов;

– сервис обработки и валидации данных заявлений. Этот сервис будет

отвечать за обработку входных данных заявлений. Он может выполнять предварительную валидацию данных (например, проверка на корректность формата данных, наличие всех необходимых документов) и подготовку заявлений к последующей проверке. В случае необходимости, сервис будет выполнять автоматическую проверку на соответствие заявлений установленным правилам и стандартам (например, наличие всех обязательных полей);

– сервис генерации отчетов и статистики. Для управления процессами заселения и распределения комнат может потребоваться регулярная генерация отчетов и статистических данных. Сервис будет собирать и агрегировать информацию о текущем статусе заявлений, распределении мест, а также статистику по договорам на проживание. Он будет генерировать подробные отчеты для дальнейшего анализа, принятия решений и подготовки внутренней документации;

– сервис расчета приоритетности заявлений. Для справедливого и эффективного распределения мест можно ввести систему приоритетов для заявлений. Сервис будет обрабатывать и вычислять приоритеты для заявлений на основе различных критериев, таких как успеваемость студентов, наличие льгот, социальный статус, а также других факторов, которые могут быть определены правилами университета или общежития. В результате, сервис будет вычислять очередность заявлений для дальнейшего рассмотрения;

– сервис синхронизации и обработки конфликтных данных. Когда происходит изменение в данных (например, изменение состава студентов, переселение, добавление новых заявок или договоров), может возникнуть необходимость в обработке конфликтных ситуаций. Сервис будет отслеживать изменения в данных и корректировать их в случае ошибок или несоответствий. Например, если студент был случайно распределен в комнату, которая уже занята, сервис будет инициировать процесс разрешения конфликта и автоматической перераспределения.

Модульный подход к проектированию приложений обеспечивает значительные преимущества в организации, читаемости и поддерживаемости кода.

Он способствует созданию гибкой и масштабируемой архитектуры, которая легко адаптируется к изменениям и расширению функциональности.

2.4 Программные платформы для разработки

Разработка многофункционального веб-приложения для управления процессами, связанными с общежитиями Амурского государственного университета, реализуется с разделением на фронтенд и бэкенд. Серверная часть (бэкенд) отвечает за обработку данных, выполнение бизнес-логики, взаимодействие с базой данных и обработку клиентских запросов.

Для реализации бэкенда выбран фреймворк Ruby on Rails, основанный на архитектурном паттерне MVC (Model-View-Controller), который структурирует приложение на три компонента: модель, представление и контроллер (рисунок 16).

Платформа Ruby on Rails обеспечивает разработчикам множество преимуществ:

– минимальная необходимость в конфигурации благодаря принципу «соглашение важнее настройки», что ускоряет старт разработки;

– встроенные инструменты, включая ORM Active Record, систему маршрутизации, шаблонизатор Action View и автоматическую валидацию моделей;

– активное сообщество, предлагающее широкий спектр библиотек, обучающих материалов и быструю техническую поддержку.

Наряду с этим, разработка по данной технологии будет вестись на языке программирования Ruby. Это интерпретируемый, объектно-ориентированный язык, созданный Юкихиро Мацумото в середине 1990-х годов. Ruby был разработан с упором на простоту и удобство использования, с акцентом на ясность, читаемость и выразительность кода, что особенно важно в условиях быстрой итерационной разработки.

Одной из ключевых особенностей Ruby является гибкая синтаксическая структура, которая позволяет программистам писать код, близкий к естественному языку. Такая особенность делает язык особенно привлекательным для быстрого прототипирования и реализации сложной логики при минимальных

издержках на поддержание кода. Ruby поддерживает несколько парадигм программирования, включая объектно-ориентированную, функциональную и императивную, что делает его универсальным инструментом для решения широкого спектра задач.

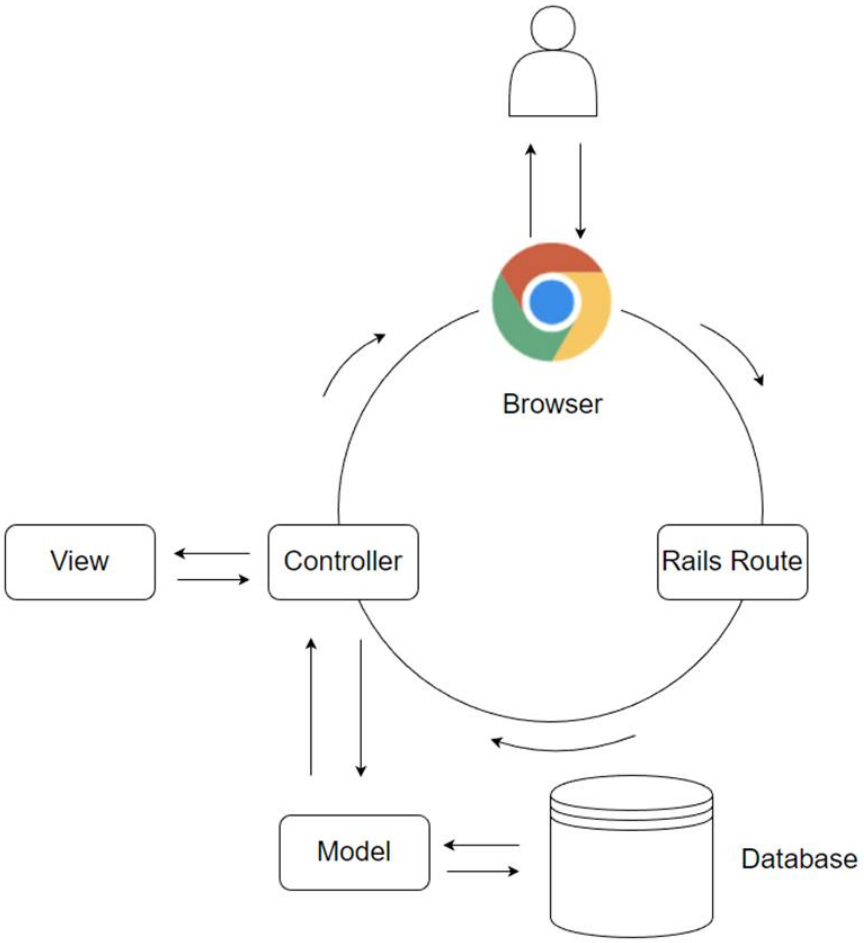

Рисунок 16 – Архитектурный паттерн MVC

Выбор MVC оправдан в ряде случаев по следующим причинам:

– разделение обязанностей. Модель (Model) отвечает за управление данными и бизнес-логику. Представление (View) отвечает за отображение данных пользователю. Контроллер (Controller) управляет взаимодействием между Моделью и Представлением, обрабатывает пользовательские действия. Это разделение позволяет сделать код более организованным и упрощает его сопровождение, так как изменение в одном компоненте не требует изменения других компонентов.

– модульность. Каждая часть MVC может разрабатываться и тестироваться отдельно, что делает приложение более модульным и простым в сопровождении. Например, можно изменить логику отображения данных в Представлении, не затрагивая бизнес-логику в Модели;

– паттерн MVC обеспечивает удобство масштабирования и поддержку проекта. Чёткое разделение компонентов позволяет легко добавлять новые функции, сохраняя структуру приложения понятной и управляемой. Кроме того, изоляция логических слоёв упрощает процесс тестирования. Модель, представление и контроллер можно проверять независимо друг от друга, что повышает надёжность и стабильность системы в целом;

– повторное использование кода. Модель может использоваться повторно в нескольких Представлениях или даже в разных проектах, что помогает избегать дублирования кода;

– четкая структура. MVC предоставляет разработчикам четкую структуру для организации кода, что помогает новым участникам команды быстрее разобраться в проекте;

– поддержка современных фреймворков. Многие популярные фреймворки (например, ASP.NET, Django, Ruby on Rails) реализуют MVC паттерн, что делает его выбор особенно привлекательным для использования в веб-разработке и других областях.

Для надёжного хранения и обработки данных в системе используется объектно-реляционная СУБД PostgreSQL. Она была разработана на факультете компьютерных наук Калифорнийского университета в Беркли и стала одной из самых передовых в своём классе. PostgreSQL заложила основу для множества современных функций, которые впоследствии появились и в коммерческих СУБД.

Система управления базами данных PostgreSQL поддерживает расширенный стандарт SQL и предлагает следующие ключевые возможности:

– сложные запросы позволяют формировать выборки с использованием вложенных условий, подзапросов, агрегатных функций и оконных выражений, что делает анализ данных более гибким и мощным;

– внешние ключи обеспечивают целостность связей между таблицами, не позволяя создать или изменить записи, нарушающие логическую структуру базы;

– триггеры автоматически выполняют заданные действия при изменении данных (например, при вставке, обновлении или удалении), что удобно для реализации бизнес-правил на уровне базы;

– обновляемые представления позволяют обращаться к виртуальным таблицам, созданным на основе запросов, и при необходимости вносить в них изменения, синхронизируемые с основными таблицами;

– целостность транзакций гарантирует, что все операции в рамках транзакции будут выполнены полностью либо не будут выполнены вовсе, сохраняя непротиворечивое состояние данных;

– многоверсионность (MVCC) обеспечивает высокую производительность при параллельной работе нескольких пользователей, позволяя читать и изменять данные без блокировок и задержек.

Взаимодействие с базой данных PostgreSQL в фреймворке Rails осуществляется через встроенный ORM – ActiveRecord. Инструмент реализует объектно-ориентированный подход к работе с данными, существенно упрощая процесс разработки и минимизируя необходимость ручного написания SQL-запросов.

Связь между приложением и базой данных в современных веб-приложениях, в том числе разработанных с использованием фреймворка Ruby on Rails, настраивается через специальный конфигурационный файл. В этом файле указываются все необходимые параметры для успешного подключения к базе данных: её имя, имя пользователя, пароль для доступа, а также адрес хоста, на

котором размещена база. Параметры позволяют приложению установить стабильное соединение и взаимодействовать с базой данных без необходимости ручного вмешательства.

После того как соединение установлено, разработчик получает возможность работать с базой данных при помощи моделей, представляющих собой объектно-ориентированное представление таблиц в базе. Каждая модель соответствует определённой таблице и позволяет взаимодействовать с данными более удобно и безопасно, без необходимости напрямую писать SQL-запросы. Модели в Rails наследуются от базового класса ApplicationRecord, что автоматически предоставляет им базовую функциональность. Внутри моделей задаются ассоциации (например, связи «один ко многим», «многие ко многим» и т.п.), которые описывают, как различные таблицы в базе данных взаимодействуют друг с другом. Также в моделях прописываются правила валидации, проверяющие корректность данных перед их сохранением в базу, например, наличие обязательных полей, уникальность значений или правильный формат.

Для управления структурой базы данных в Rails применяются миграции – специальные файлы, содержащие инструкции по созданию и изменению таблиц, полей, индексов и других элементов схемы базы данных. Миграции позволяют добавлять и удалять таблицы, изменять структуру уже существующих, задавать ограничения на уровне схемы, а также создавать связи между таблицами. Все изменения вносятся в виде кода и сохраняются в истории проекта, что даёт разработчикам возможность отслеживать, повторять или отменять изменения при необходимости.

ActiveRecord предоставляет богатый набор методов для выполнения операций с данными: find, where, create, update, destroy и многие другие. Методы позволяют гибко фильтровать, обновлять и удалять записи, при этом генерируемые SQL-запросы формируются автоматически. Также ActiveRecord поддерживает ассоциации между моделями: «один к одному», «один ко многим» и «многие ко многим», что позволяет устанавливать логические связи

между таблицами и обращаться к связанным данным без сложных SQL-запросов. В случае необходимости разработчик может использовать чистый SQL, что особенно полезно при создании сложных запросов с высокой производительностью. Для оптимизации работы базы данных PostgreSQL предлагает инструменты вроде индексов, анализа запросов и тонкой настройки плана выполнения.

Веб-приложение взаимодействует с серверной частью с помощью REST API, который обеспечивает стандартизированный обмен данными между клиентом и сервером (см. рисунок 17). Такой подход делает архитектуру гибкой, масштабируемой и легко расширяемой. Основу REST составляют следующие ключевые принципы:

– отделение клиента от сервера (Client–Server). Логика клиентского интерфейса и логика обработки данных разделены. Это позволяет независимо развивать клиентскую часть (интерфейс) и сервер (бизнес-логику и базу данных), а также упрощает сопровождение системы;

– единый интерфейс (Uniform Interface). Все взаимодействие с сервером происходит через единый набор стандартных запросов (GET, POST, PUT, DELETE и др.). Такой подход упрощает понимание API, делает его предсказуемым и универсальным для различных клиентов (веб, мобильных и других приложений);

– кэшируемость (Cacheable), данные, полученные от сервера, могут кэшироваться на стороне клиента, что позволяет снижать нагрузку на сервер, ускорять отклик системы и повышать общую производительность, особенно при повторных запросах к неизменным данным.

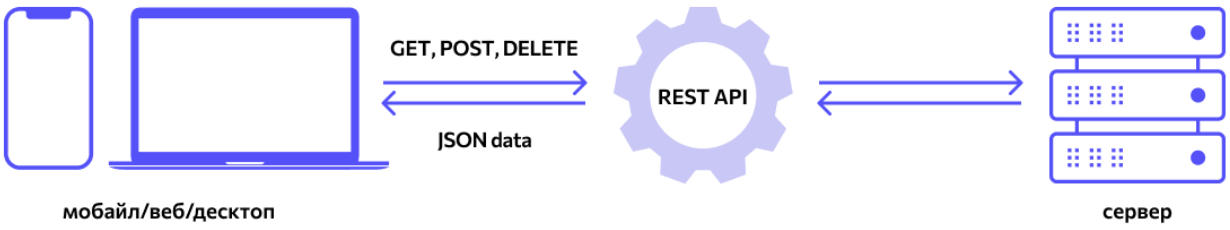

Рисунок 17 – Клиент серверная архитектура

Использование REST API является обоснованным выбором для разработки современных веб-приложений и систем, так как оно основывается на архитектурном стиле, который обеспечивает простоту и гибкость взаимодействия между различными компонентами системы. Основное преимущество REST заключается в том, что он использует стандартные протоколы передачи данных, такие как HTTP, что делает его совместимым с широким спектром устройств и платформ. Это позволяет системам, написанным на разных языках программирования, легко взаимодействовать между собой.

REST API способствует созданию масштабируемых и легко поддерживаемых приложений. Важно отметить, что взаимодействие в REST построено на основе простых запросов, которые могут быть легко реализованы и поддержаны. Это особенно полезно при работе с распределенными системами или микросервисной архитектурой, где отдельные компоненты системы должны эффективно обмениваться данными.

На стороне первой части веб-приложения «фронтэнде» используется фреймворк Vue.js, основным языком программирования для разработки является JavaScript. Vue.js – это прогрессивный фреймворк для создания пользовательских интерфейсов и одностраничных приложений. Его основная цель – обеспечить простоту интеграции с другими проектами и библиотеками, а также предоставить удобные средства для разработки сложных приложений.

Одной из главных особенностей Vue.js является его реактивная система. Она позволяет отслеживать изменения в данных и автоматически обновлять DOM, что упрощает работу с интерфейсом и делает код более читаемым. Благодаря этому, разработчики могут сосредоточиться на логике приложения, не беспокоясь о сложностях манипуляций с DOM.

Vue.js также имеет легковесный и гибкий дизайн. Его ядро содержит лишь основные функции, что обеспечивает быструю загрузку и эффективность. При этом он легко расширяется с помощью официальных и сторонних плагинов, таких как Vue Router для маршрутизации и Vuex для управления

состоянием. С точки зрения синтаксиса, Vue.js предлагает декларативный подход к созданию интерфейсов с использованием шаблонов, основанных на HTML. Это делает код интуитивно понятным, особенно для тех, кто уже знаком с HTML и JavaScript. Кроме того, Vue.js поддерживает компоненты, что способствует модульности и переиспользованию кода.

Рисунок 18 иллюстрирует общую структуру взаимодействия всех используемых технологий и решений, обобщая ранее рассмотренные программные компоненты. На клиентской стороне (Frontend) используется связка современных JavaScript-технологий, включая Vue.js как основной фреймворк для построения интерфейса, Vuetify для визуального оформления на основе Material Design, Vuex для управления состоянием приложения, а также Axios для выполнения HTTP-запросов. Все компоненты фронтенда объединяются и компилируются с помощью сборщика Webpack.

Серверная часть (Backend) реализована на базе фреймворка Ruby on Rails, обеспечивающего обработку бизнес-логики, маршрутизацию и работу с базой данных. В качестве системы управления базами данных используется PostgreSQL. Взаимодействие между клиентом и сервером осуществляется по принципам REST API: клиент направляет HTTP-запросы (GET, POST, PUT, DELETE), а сервер возвращает ответы в форматах JSON, XML или HTML.

Рисунок 18 – Общая структура взаимодействия используемых технологий и решений

Vue.js также известен своей простой интеграцией с другими библиотеками и существующими проектами. Это позволяет внедрять Vue.js в проекты постепенно, без необходимости полной переписки кода. Также Vue.js имеет отличную документацию и активное сообщество, что облегчает процесс обучения и поиска решения возникающих проблем.

Другим преимуществом Vue.js является его производительность. Фреймворк использует виртуальный DOM, что минимизирует количество операций по обновлению реального DOM и, таким образом, повышает общую эффективность приложения. Кроме того, Vue.js предлагает оптимизированные средства для обработки данных и асинхронных операций.

2.5 Архитектура данных и принципы ее построения

Структура базы данных предназначена для автоматизации процесса заселения людей в общежитие. Она охватывает все ключевые этапы, начиная с подачи заявлений и их рассмотрения комиссией, до распределения по подразделениям, формирования приказов на заселение, заключения договоров и ведения истории всех этапов взаимодействия, общий вид базы данных представлен на рисунке 19.

Основой системы является таблица «StatementHostels», в которой фиксируются поданные заявления. В ней хранятся данные о заявителе, типе заселения, выбранном месте проживания и статусе заявления. Важно, что каждая заявка остается в системе независимо от ее текущего состояния, что позволяет вести историю всех действий и анализировать процесс заселения. Связь с таблицей «People» дает возможность привязать каждое заявление к конкретному человеку.

После подачи заявления оно может быть рассмотрено комиссией, что отражается в таблице «CommissionStmHostelStatuses». Здесь фиксируется идентификатор заявления, информация о человеке, принимающем решение, статус

рассмотрения и дата принятия решения. Такая структура позволяет отслеживать процесс утверждения или отклонения заявок и хранить историю изменений. Комментарии комиссии записываются в таблице «StmHostelCommissionComments», что дает возможность объяснять причины отказов или фиксировать дополнительные требования.

В случае одобрения заявления происходит его распределение по подразделениям, что реализуется через таблицу «DistributionPersonDepartments». В ней хранится информация о том, к какому подразделению прикреплена заявка, а также ссылка на приказ на заселение, который фиксируется в таблице «HostelOrders».

Благодаря продуманной структуре хранения и обработки данных система не только организует процесс распределения мест, но и делает его максимально прозрачным, управляемым и предсказуемым. Она обеспечивает строгий контроль за каждым этапом – от подачи заявления до окончательного распределения. Подход позволяет избежать повторной подачи одинаковых заявлений, минимизирует ошибки, исключает дублирование и предотвращает стихийное или нерегламентированное заселение студентов. В результате улучшается эффективность работы комиссии и повышается удовлетворённость пользователей системой.

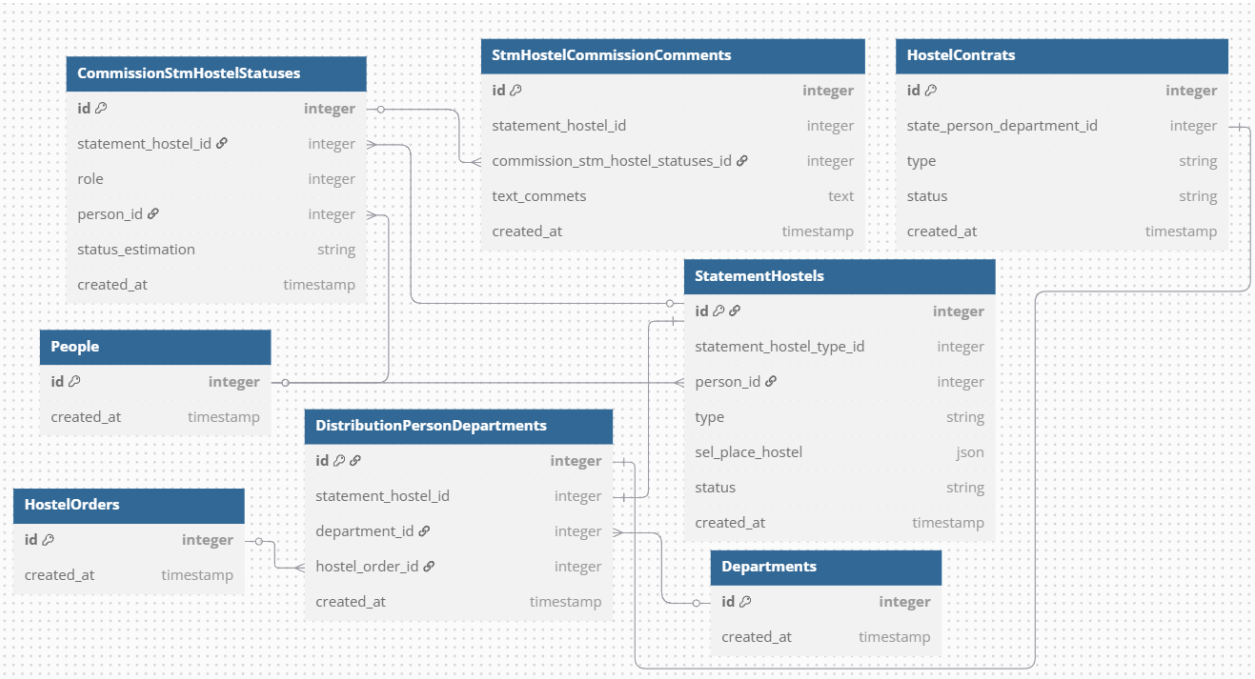

Рисунок 19 – Структура база данных

После распределения создается договор, информация о котором хранится в таблице «HostelContrats». Здесь фиксируется связь между заявителем и подразделением, а также тип договора и его статус. Это необходимо для управления условиями проживания и отслеживания их изменений.

Предложенная схема базы данных обеспечивает четкую организацию процесса заселения, исключая дублирование информации и упрощая обработку заявлений на всех этапах – от подачи заявки до заключения договора. Благодаря установленным связям между таблицами система гарантирует целостность данных: каждое заявление привязывается к конкретному студенту, проходит проверку комиссией, получает статус распределения в общежитие и сопровождается соответствующими документами, такими как приказ о заселении и договор.

Такая структура позволяет формировать детализированные отчеты, включая информацию о количестве поданных и одобренных заявлений, статусах рассмотрения, загрузке общежитий и заключенных договорах. Например, можно отслеживать процент одобренных заявок по каждому подразделению, анализировать причины отказов или выявлять факторы, влияющие на скорость заселения.

Ключевым преимуществом предложенной модели является возможность масштабирования без необходимости внесения существенных изменений. В случае изменения правил заселения можно добавить новые статусы заявок в таблице «StatementHostels», расширить категории договоров в «HostelContrats» или интегрировать систему с внешними учетными сервисами, такими как Контур.Сайн для цифрового подписания документов. Благодаря четкой связи между сущностями система остается гибкой и адаптируемой к изменениям нормативных требований и новых бизнес-процессов.

2.6 Цифровая идентификация и подписание договора о заселении

В процессе подписания договора о заселении в общежитие используется сервис электронной подписи Контур.Сайн, который обеспечивает юридически значимое подписание документа в цифровом формате.

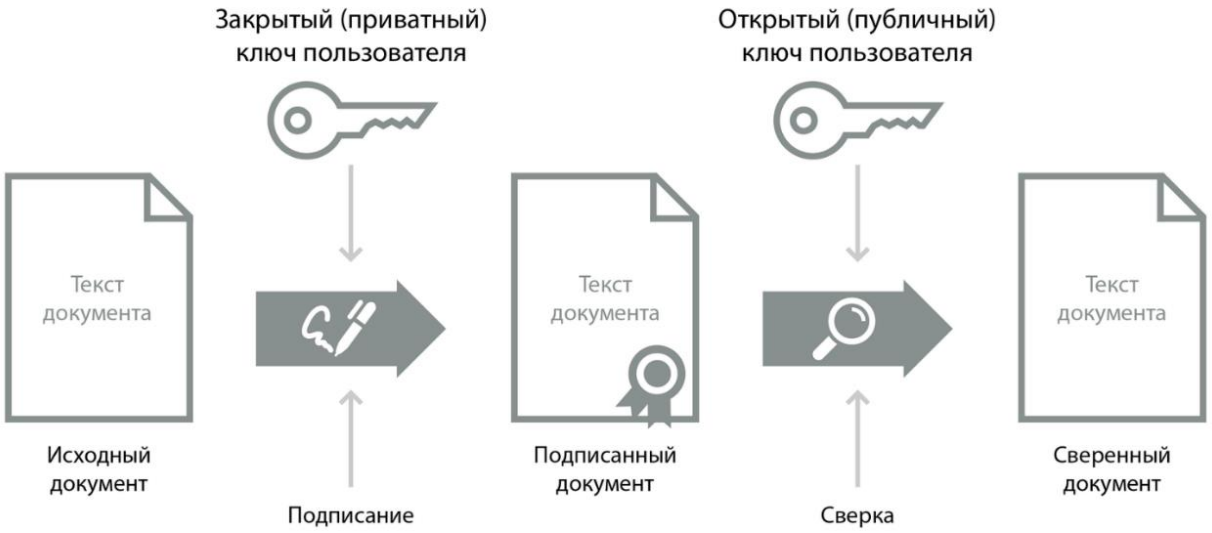

Рисунок 20 – Формирование ЭЦП

Студент проходит процедуру аутентификации через Госуслуги (ЕСИА), что гарантирует надежную идентификацию личности и исключает возможность подлога. После успешной авторизации ему предоставляется доступ к сформированному договору, который необходимо подписать.

Процесс подписания реализуется по принципу одного клика: студент подтверждает свое согласие с условиями договора, после чего система инициирует процесс криптографического наложения усиленной квалифицир-ованной электронной подписи (УКЭП). Подписание выполняется в соответствии с требованиями Федерального закона № 63-ФЗ «Об электронной подписи», что придает документу полную юридическую силу.

Контур.Сайн автоматически формирует криптографически стойкое хэш-значение документа и подписывает его закрытым ключом пользователя, хранящимся в защищенном хранилище Контура. Сформированная квалифицированная электронная подпись (КЭП) вместе с подписанным файлом передается в систему. После успешного завершения процесса подписания договор фикси-

руется в базе данных с присвоением ему юридически значимого статуса «Подписан» и сразу же становится доступным для дальнейшего использования. Сразу после этого происходит автоматическая передача подписанного и юридически действительного договора в систему управления общежитием по заранее настроенному интеграционному сценарию.

Благодаря этому подписанный договор мгновенно поступает в работу ответственным сотрудникам общежития. Весь процесс подписания, фиксации и передачи документа выполняется системой автоматически, обеспечивая максимальную оперативность и исключая ручные ошибки.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

3.1. Структурное описание реализации модулей системы

Данный раздел содержит структурное описание реализации модулей системы. Описание основано на ключевых компонентах архитектуры: контроллерах, моделях, сервисном слое и представлениях. Контроллеры управляют обработкой запросов и определяют поведение системы в ответ на действия пользователя, модели обеспечивают работу с данными, сервисный слой отвечает за вынесение сложной бизнес-логики за пределы контроллеров, а представления формируют пользовательский интерфейс.

3.1.1 Модуль распределения комнат по факультетам

При разработке данного модуля можно выявить два основных процесса: процесс инициализации модуля «Распределение мест в общежитии»; процесс распределения мест общежитий по факультетам.

Для более детального анализа были разработаны соответствующие диаграммы состояний (Рисунок 21, Рисунок 22), показывающие взаимодействие и переход между ключевыми состояниями.

На рисунке 1 показаны состояния процесса распределения мест в общежитиях по факультетам и переходы между ними:

– извлечение данных из БД – начальное состояние, в котором модуль отправляет запрос к базе данных для получения актуальной информации о комнатах в общежитии. При успешном ответе данные о комнатах передаются на следующий этап. Если возникает ошибка или неожиданный ответ, процесс завершается с ошибкой;

– агрегирование полученных данных – на этом этапе происходит группировка данных по факультетам и общежитиям. Выполняется проверка данных для обеспечения их целостности и корректности, и результат передаётся на следующие шаги в виде структурированных данных;

– подготовка записей соотношения мест общежитий с факультетами – система создаёт и структурирует записи, которые фиксируют распределение мест между факультетами и общежитиями. Эти записи готовятся для последующего использования в модуле распределения;

– формирование статистики по общежитиям и факультетам – система производит подсчёт свободных и занятых мест по каждому общежитию и факультету. Этот шаг позволяет собрать сводную информацию для анализа доступности мест;

– завершение инициализации основных данных – финальное состояние, в котором система завершает процесс инициализации модуля, обеспечивая готовность данных для дальнейшего использования.

Рисунок 21 – Диаграмма состояний процесса инициализации модуля

Далее на втором рисунке описывается процесс распределения мест в общежитиях по факультетам, состоящий из нескольких этапов, от инициализации данных до сохранения обновленной информации о местах:

– инициализация записи соотношения мест. Процесс начинается с открытия формы для факультета, после чего инициализируется создание записей о соответствии комнат общежитий и факультета. На этом этапе определяется порядок добавленных комнат и количество мест, а также вычисляется статус каждого места;

– извлечение актуальных данных о комнатах из 1С. Для получения информации о комнатах система выполняет SOAP-запрос к 1С, откуда возвращаются актуальные данные о комнатах. Эти данные включают информацию о наличии мест, распределении по секциям и комнатам, что необходимо для последующей обработки;

– агрегирование полученных данных. Данные, полученные из 1С, агрегируются и структурируются по общежитиям, секциям и комнатам. Выполняется проверка данных для обеспечения корректности и целостности информации, что важно для точного распределения мест по факультетам;

– закрепление комнаты (Обновление данных). Пользователь выбирает нужные комнаты для распределения их за факультетом. Данные о выбранных комнатах структурируются и направляются на обновление записей;

– обновление комнат для факультета. На этом этапе данные проходят валидацию и сохраняются в базе данных. Процесс завершается закрытием формы, что сигнализирует о завершении обновления и распределения мест по факультету.

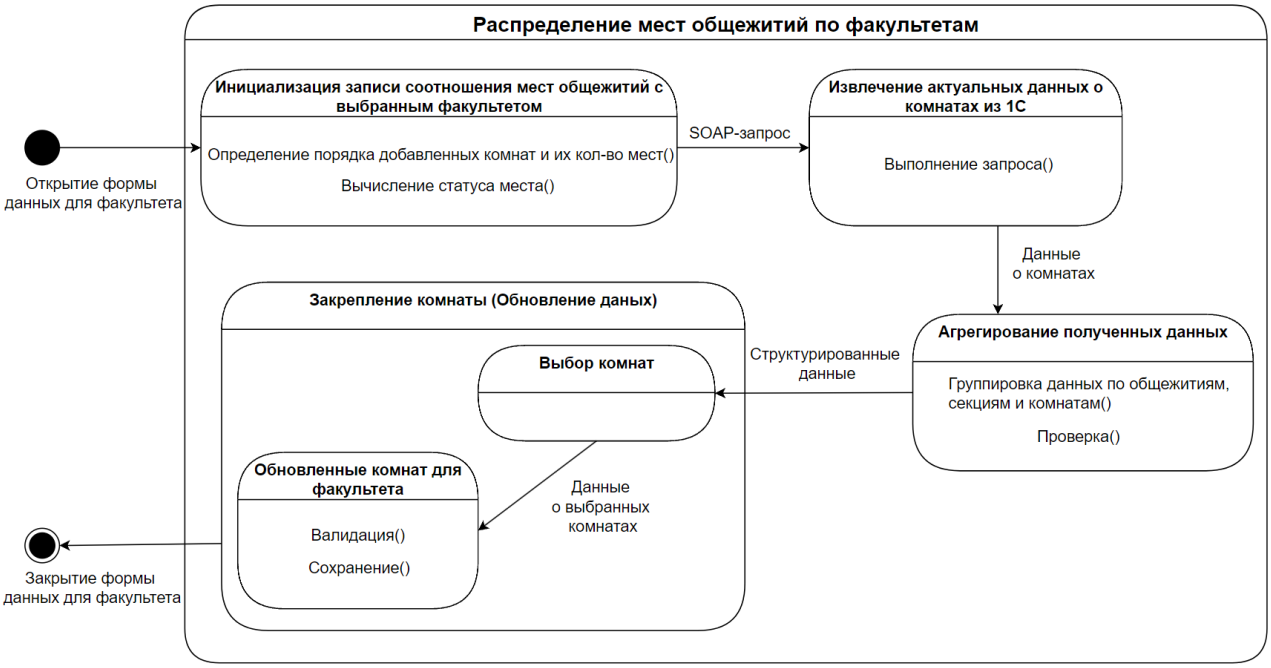

Рисунок 22 – Диаграмма состояний процесса распределения мест общежитий по факультетам

Далее диаграмма компонентов, изображенная на рисунке 23, представляет реальную архитектуру модуля управления распределением мест в общежитиях с интеграцией данных из системы «1С-Университет». Она разделена на три основные части: Сервер базы данных (БД), Web-сервер (Backend) и Web-браузер (Frontend).

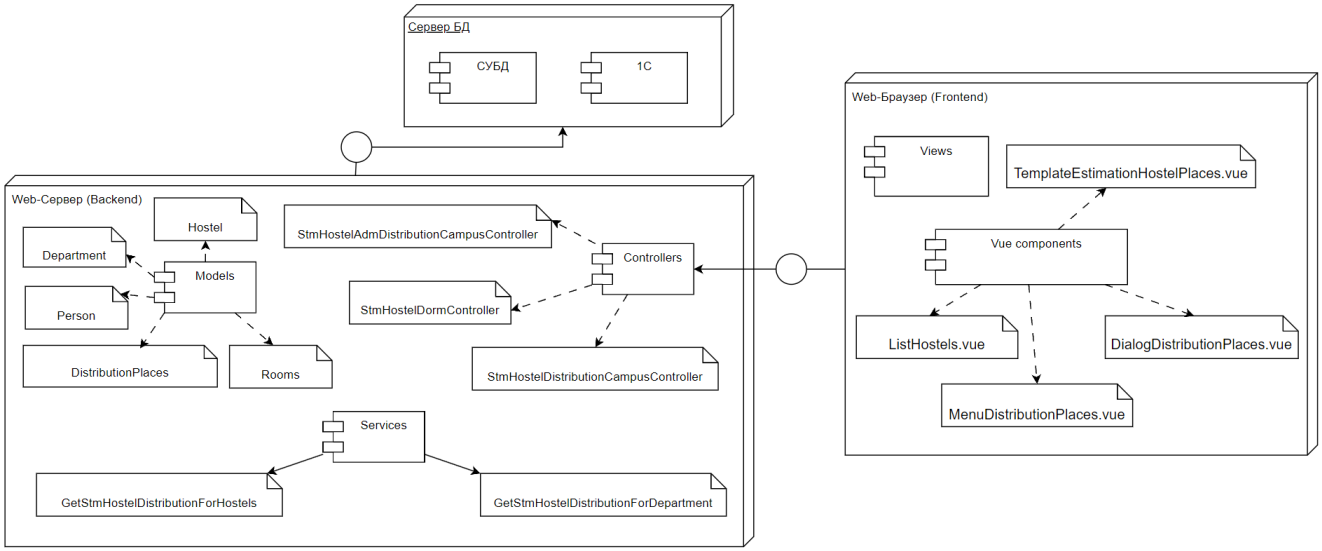

Рисунок 23 – Диаграмма компонентов модуля распределения комнат по факультетам

Процессы передачи и обработки данных реализуются в рамках трех классов-контроллеров, описанных более подробней ниже.

Контроллер StatementHostel::Distribution::CampusAndFacultiesController отвечает за управление процессом распределения мест в общежитиях между факультетами в системе. Он предоставляет административные функции для создания, обновления, отображения и удаления данных о закрепленных местах за факультетами. Доступ к действиям контроллера ограничен только для пользователей с правами администратора. Контроллер обрабатывает запросы на получение списка свободных мест, обновление информации о комнатах, а также получение статуса распределения как по кампусам, так и по факультетам через соответствующие сервисы. В основе работы лежат модели FacultyPlace и Department, а также бизнес-логика, вынесенная в сервисные

классы. Реализация включает защиту от неавторизованного доступа и проверку наличия записей перед их созданием или обновлением, что обеспечивает корректное и безопасное управление распределением мест.

Таблица 1 – Описание класса «StatementHostel::Distribution::CampusAndFacultiesController»

Элемент	Описание
1	2
Класс	StatementHostel::Distribution::CampusAndFacultiesController
Наследование	Наследуется от ApplicationController
Layout	Использует layout 'application'
Фильтры	- before_action :authenticate_user! – аутентификация пользователя - before_action :valid_user? – проверка на админа
Права доступа	Только для администратора (current_user.admin?)
Методы контроллера	
index	Отображает страницу распределения общежитий (statement_hostel/index_distribution)
create	Создаёт запись о распределении мест по факультету, если нет уже существующей записи
show	Показывает структурированную информацию по факультету (faculty_places) в формате JSON
update	Добавляет новые комнаты к факультету
destroy	Удаляет выбранные комнаты у факультета, при отсутствии комнат удаляет всю запись
get_free_place	Получает свободные места в общежитиях через сервис
update_room	Обновляет количество мест в комнате (info["total"])
status_places	Получает статусы общежитий (кампусов) через сервис

Продолжение таблицы 1

status_faculties_places	Получает статусы факультетских мест через сервис
Приватные методы	valid_user? – проверка, что пользователь – администратор, иначе редирект на главную страницу
Зависимости	- Модель FacultyPlace - Модель Department - Сервисы DepartmentServices::Hostel:: StatusesHostelPlacesOrFaculties; DepartmentServices::Hostel::GetFreePlaces

Были реализованы дополнительные классы для формирования статистических данных, а также для вспомогательных представлений структуры комнат общежитий.

Контроллер StatementHostel::Distribution::StmHostelDormController отвечает за получение и отображение структуры распределения общежитий по факультетам. Перед выполнением действий он требует аутентификацию пользователя и включает модуль StatementHostel::StatementHostelApi, что позволяет использовать дополнительные методы и функциональность. Основной метод index через сервис DepartmentServices::Hostel::GetStructureHostel получает карту распределения общежитий по факультетам для текущего департамента и возвращает данные в формате JSON. В случае возникновения ошибки возвращается сообщение об ошибке, обеспечивая базовую обработку исключений и стабильность работы контроллера.

Контроллер StatementHostel::Distribution::StmHostelPreviewDistributionController предназначен для формирования и предоставления предварительного списка распределённых заявлений на места в общежитии. Он требует аутентификацию пользователя и включает модуль StatementHostel::StatementHostelApi, а также отключает проверку CSRF-токена. Основной метод index формирует выборку заявлений с использова-

нием сложных объединений между моделями StatementHostel, Person, Department и связанными сущностями, отбирая по каждому студенту только наиболее актуальное заявление, исключая определённые типы заявлений и фильтруя их по факультету пользователя. Результат передаётся в формате JSON для дальнейшего отображения или обработки.

3.1.2 Модуль подачи заявления

Модуль предназначен для автоматизации процесса подачи заявлений на предоставление места в общежитии в вузе через личный кабинет студента. Он ориентируется на данные студента, включая личную информацию, академическую успеваемость, текущее место проживания и историю ранее поданных заявлений. На основе этих данных модуль проверяет условия допуска к подаче заявления в соответствии с установленными правилами вуза. При успешной проверке происходит формирование структурированных данных и инициализация формы подачи заявления, которую студент может заполнить и отправить. В случае ошибок при получении данных из базы модуль корректно завершает процесс, предотвращая возникновение сбоев и обеспечивая надежность работы личного кабинета.

Работа модуля включается в себя два ключевых состояния, описанные с помощью диаграмм, изображенных на рисунках 24 – 25.

Процесс инициализации модуля «Подача заявления» протекает по следующему порядку (Рисунок 24):

– запуск запроса. Модуль отправляет запрос в БД для получения данных студента (состояние «Выполнение запроса()»);

– результат запроса. Если запрос выполнен успешно, данные передаются на следующий этап – «Подготовка и обработка полученных данных». Если возникает ошибка или неожиданный ответ, происходит завершение инициализации без перехода к обработке данных;

– подготовка и обработка данных. Выполняется ряд операций: подсчет рейтинга академической успеваемости; проверка текущего места проживания студента; проверка условий подачи на основе имеющихся данных; проверка

наличия ранее поданных заявлений; после выполнения этих процедур формируются структурированные данные;

– вывод данных пользователя. Структурированные данные используются для вывода информации пользователю. Параллельно инициализируется форма подачи заявления. И далее после успешной подготовки формы процесс инициализации завершается.

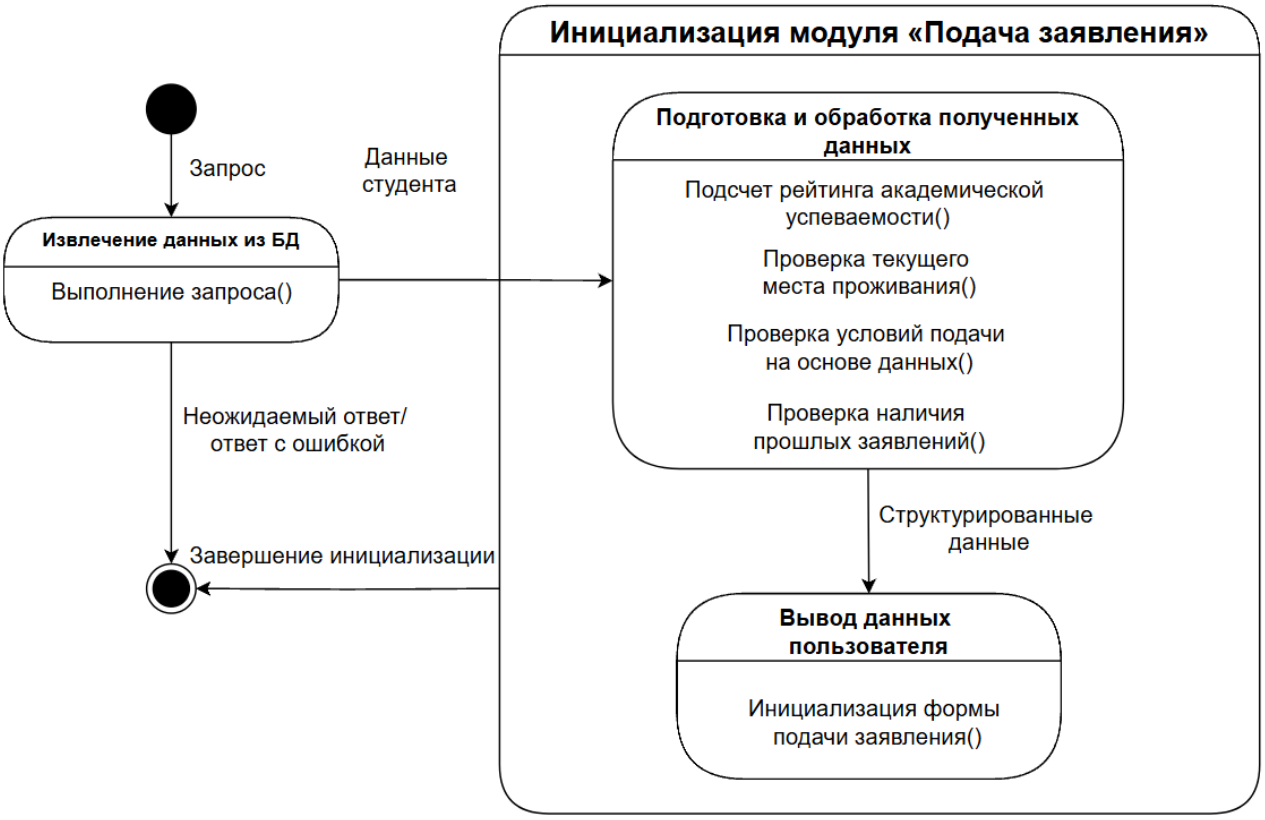

Рисунок 24 – Диаграмма состояний процесса инициализации модуля подачи заявления

Следующим состоянием является – формирование заявления. Данный процесс схематично представлен на рисунке 25 и ниже подробно описаны его этапы:

– инициализация данных пользователя. После авторизации в личном кабинете загружается базовая информация о пользователе из базы данных. Компонент отправляет запрос через API (axios) для заполнения формы заявления данными, такими как имя студента, факультет, и текущие предпочтения;

– выбор типа заявления. Соответствующий компонент отображает выпадающий список типов заявлений. Данные о доступных типах получаются с помощью асинхронного API-запроса. После выбора типа запускается валидация формы;

– выбор общежития и комнаты. Список доступных общежитий, секций и комнат фильтруется на сервере и отправляется в виде JSON для отображения во Vue2-компоненте. Пользователь выбирает общежитие, секцию и комнату через интерактивные элементы интерфейса;

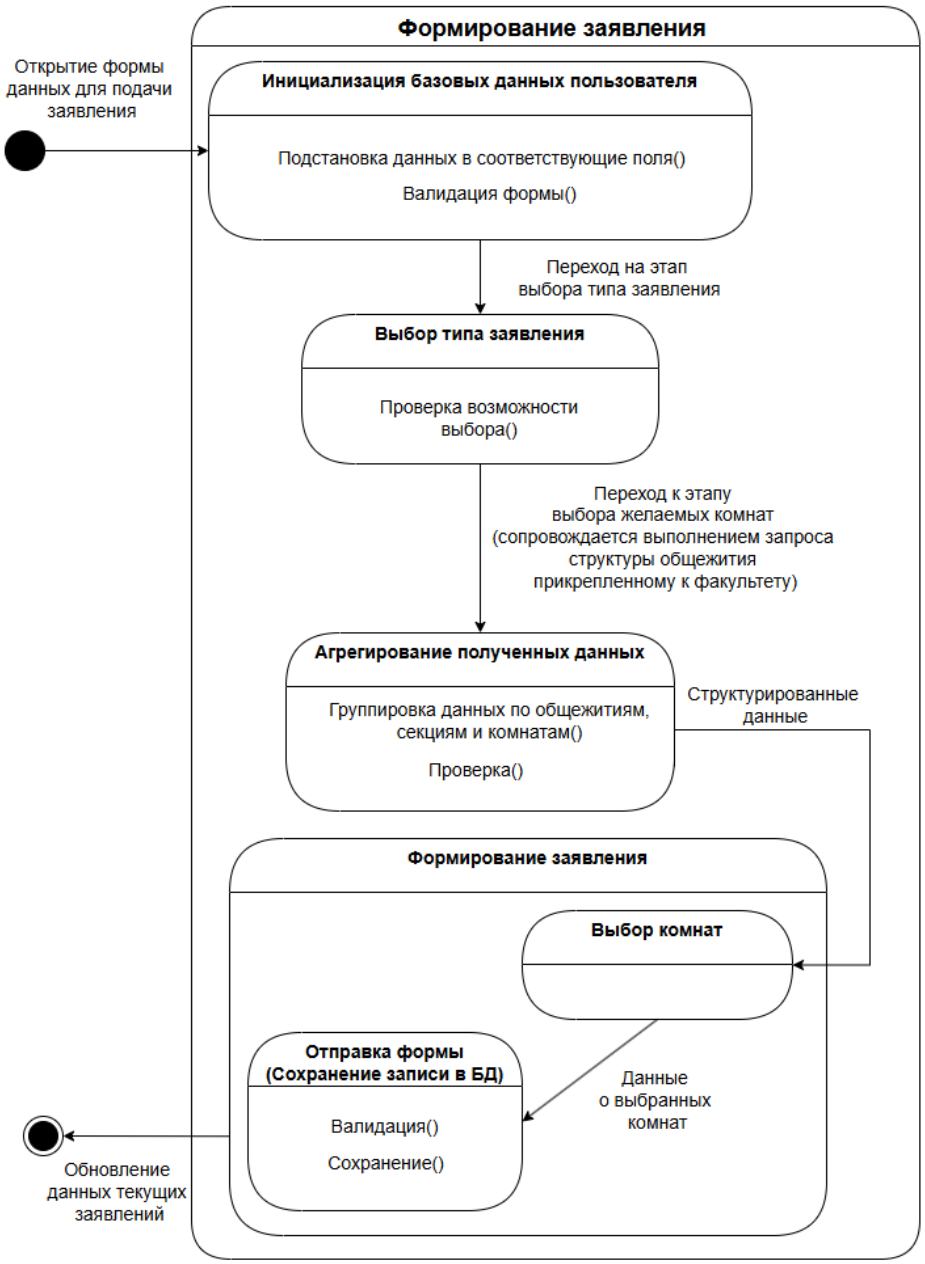

Рисунок 25 – Диаграмма процесса формирования заявления

– агрегация данных. После выбора пользователем комнаты происходит обработка данных на клиентской стороне. Frontend собирает все введённые и выбранные данные в единый объект. Передача данных на сервер через POST-запрос с использованием axios;

– формирование заявления. Rails backend принимает данные, проводит серверную валидацию. Если валидация успешна, данные сохраняются в базу. Ответ сервера возвращает статус выполнения операции;

– обновление или просмотр заявления. Если заявление уже создано, оно отображается в личном кабинете с возможностью редактирования. Rails предоставляет данные заявления через REST API;

Диаграмма компонентов на рисунке 26 иллюстрирует архитектуру модуля подачи заявлений на проживание в общежитиях вуза. Структура этого модуля состоит из трех основных частей, которые были описаны ранее в предыдущем разделе. Однако, в отличие от ранее представленных моделей, внутренние зависимости между компонентами имеют другую конфигурацию, что отражает изменения в функциональности и взаимодействии систем.

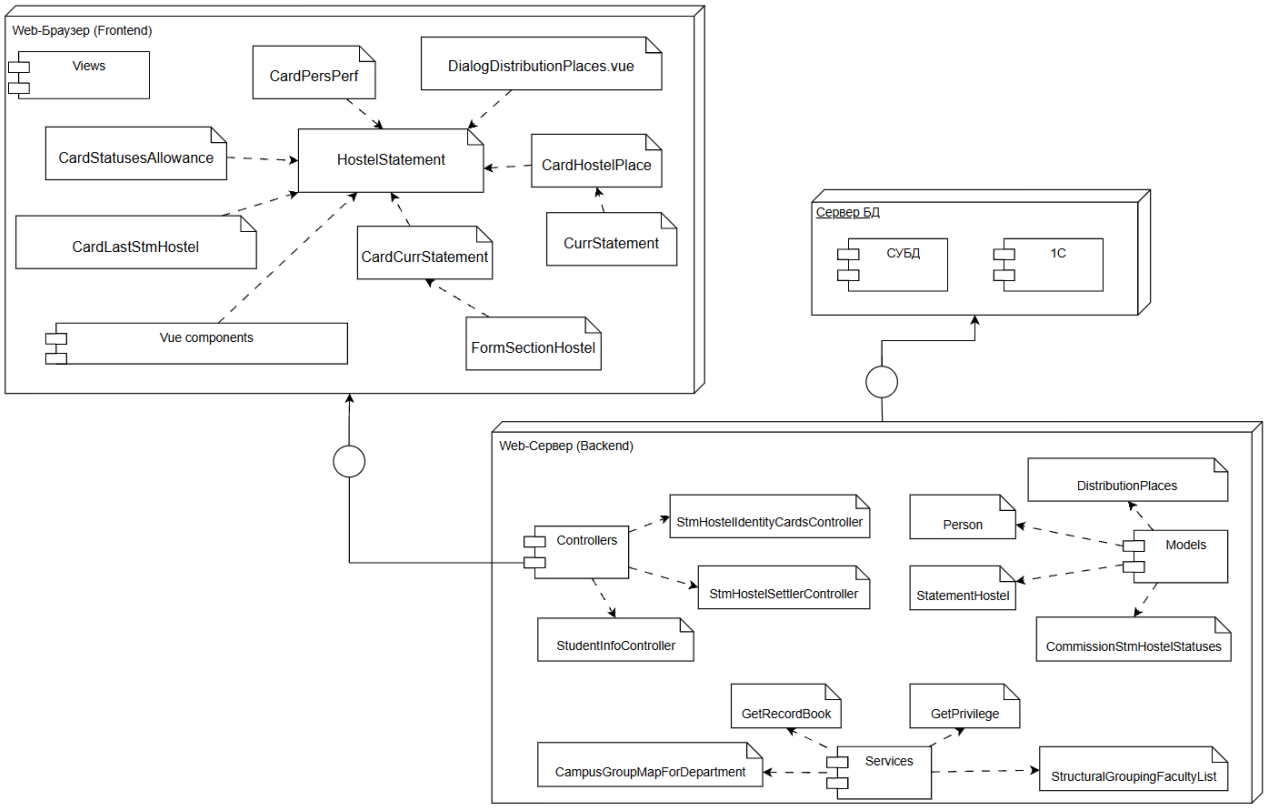

Рисунок 26 – Диаграмма компонентов модуля подачи заявления

Данный модуль разделяется на три контроллера, имеет один основной и два вспомогательных.

Класс StatementHostel::Pichers::StmHostelSettlerController отвечает за обработку заявлений студентов на заселение в общежитие. Контроллер обеспечивает создание новых заявлений с проверкой прав, рейтинга и допустимых дат, а также позволяет получать информацию о последнем активном заявлении, возможных местах заселения по факультету, обновлять существующие заявления при условии отсутствия финального подтверждения всех сторон, и удалять их при необходимости.

Внутренние проверки в системе реализованы с использованием вспомогательных методов, которые обеспечивают проверку данных и их корректность перед дальнейшей обработкой. Для эффективной работы с информацией о студенте и структурой кампуса используются сервисные слои, которые обеспечивают разделение логики обработки данных и их взаимодействие между различными частями системы. Ответы, полученные от сервисных слоёв, возвращаются в формате JSON, что позволяет легко передавать и обрабатывать данные на стороне клиента.

Таблица 2 – Описание класса «StatementHostel::Pichers::StmHostelSettlerController»

Элемент	Описание
1	2
Класс	StatementHostel::Pichers::StmHostelSettlerController – контроллер для управления процессом подачи, редактирования, удаления и получения заявлений на заселение студентов в общежития.
Наследование	Наследуется от базового класса ApplicationController.
Layout	Используется общий макет 'application'.
Фильтры	before_action :authenticate_user! – проверка аутентификации пользователя перед выполнением действий. skip_before_action :verify_authenticity_token – отключение проверки CSRF-токена.

Продолжение таблицы 2

Права доступа	Доступ к методам контроллера имеют только аутентифицированные пользователи. Нет явной проверки на уровне ролей или дополнительных ограничений внутри методов.
Методы контроллера	create – создание нового заявления с привязкой к студенту, проверкой данных и инициализацией статусов комиссии. show – получение последнего незавершённого заявления пользователя. places_for_department – получение доступных мест для факультета по department_id. last_actual_statement – получение последнего подтверждённого заявления. update – обновление заявления при соблюдении условий. destroy – удаление заявления, если оно ещё не прошло полное согласование.
Приватные методы	create_commission_stm_hostel_statuses – создание записей о статусах проверки заявления комиссией. check_change – проверка, можно ли редактировать заявление. check_select_places – валидация выбранных пользователем мест. check_period – проверка срока подачи заявления.
Зависимости	Модули: StatementHostel::StatementHostelApi. Сервисные объекты: StmHostelServices::StudentInfo::GetRecordBook, StmHostelServices::StudentInfo::GetPrivilege, DepartmentServices::Hostel::CampusGroupMapForDepartment, DepartmentServices::Hostel::StructuralGroupingFacultyList. Модели: StatementHostel, Person, CommissionStmHostelStatus.

Класс контроллера StatementHostel::Pichers::StudentInfoController управляет запросами, связанными с информацией о студентах. Контроллер использует стандартный layout «application» и требует аутентификации пользователя перед выполнением действий через before_action :authenticate_user!. Включение модуля StatementHostel::StatementHostelApi позволяет использовать API для взаимодействия с данными общежития.

В методе show происходит рендеринг данных в формате JSON, а в методах get_record_book и get_privilege вызываются сервисы для получения информации о студенте, таких как записи в учебной книжке и привилегии, соответ-

ственно, с последующим рендерингом в формате JSON. В случае возникновения ошибки при получении привилегий, возвращается пустое значение. Метод fluorography проверяет наличие информации о флюорографии у текущего пользователя, и, если она есть, возвращает её в формате JSON, иначе возвращает пустой объект.

Класс «StatementHostel::Pichers::StmHostelIdentityCardsController» в Rails управляет запросами, связанными с получением и созданием идентификационных карт студентов для общежития. Контроллер использует стандартный layout «application» и требует аутентификации пользователя через «before_action :authenticate_user!». Для работы с 1С API, контроллер подключает метод «conect_1c_api», который инициализирует клиент для взаимодействия с внешней системой через WSDL. В методе «get_identity_card» проверяется наличие идентификационной карты у текущего пользователя; если карта существует, она возвращается в формате JSON, иначе создаётся новая карта с использованием данных из методов «get_person_state_id» и «get_person_passport», а затем возвращается вновь созданная карта. Методы «get_person_state_id» и «get_person_passport» делают запросы к 1С API для получения данных о государственном ID и паспорте пользователя, соответственно. Все эти действия происходят с использованием клиента Savon для SOAP-запросов.

3.1.3 Модуль проверки заявлений

Данный модуль формируется из небольшого набора компонентов и зависимостей. Это связано с простым процессом подтверждения или отклонения заявлений участниками комиссии.

Этапы процесса проверки заявлений схематически описываются на рисунке 27:

– запрос данных. Запрос инициируется из личного кабинета, где пользователь переходит в раздел заявлений. Отправляется GET-запрос к серверу для извлечения данных из базы, содержащих актуальные заявления студента;

– извлечение данных. На серверной стороне выполняется выборка из базы данных. Например, текущие записи заявлений студентов сортируются и фильтруются. Результаты передаются клиенту в виде структурированного ответа;

– обработка данных. Полученные данные о заявлениях проходят процесс агрегации. Агрегируются статистические данные, например, о числе доступных комнат или заявлений на рассмотрении. Формируются записи заявлений с данными студента, включая тип заявления, общежитие, секцию и статус;

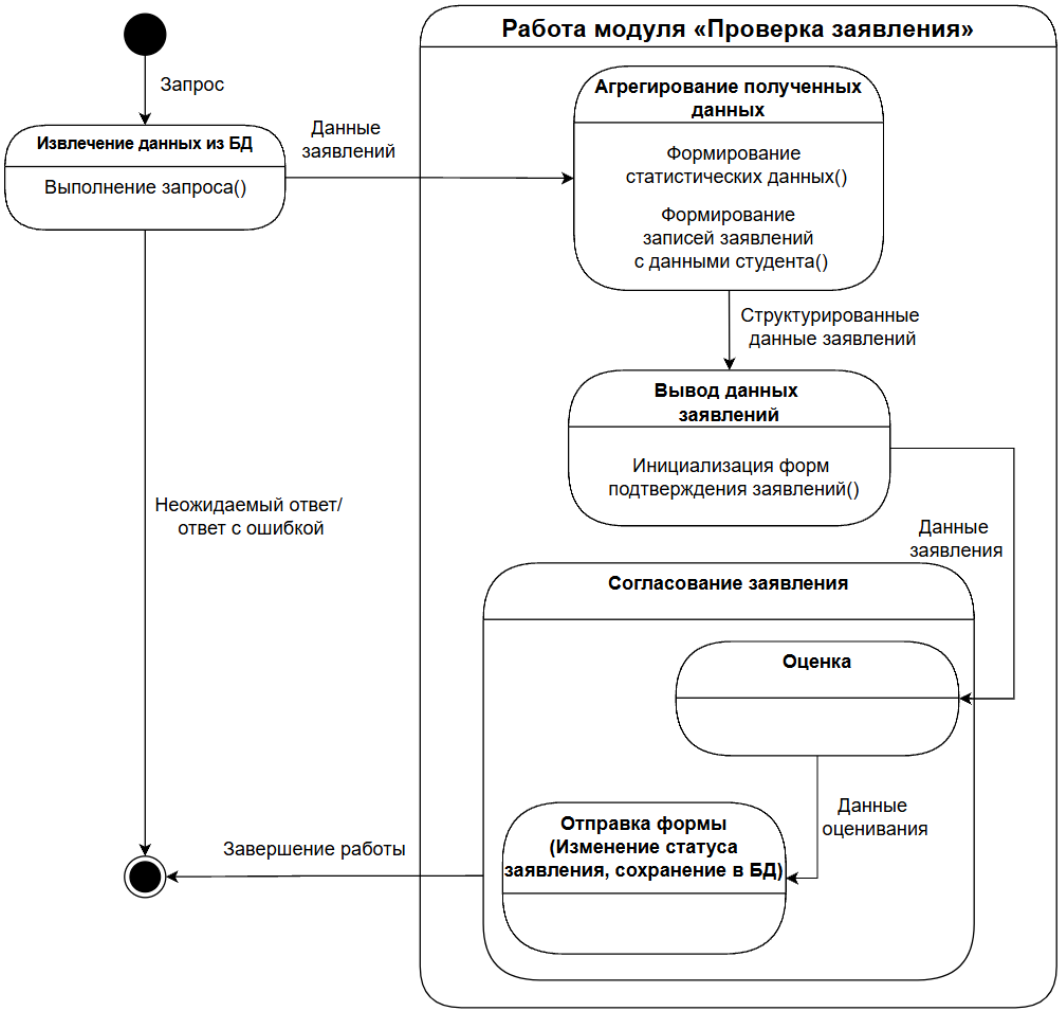

Рисунок 27 – Диаграмма состояний процесса проверки заявления

– инициализация форм. Интерфейс личного кабинета динамически создает формы подтверждения заявлений, основываясь на полученных данных. Пользователю предоставляются интерактивные элементы для проверки и согласования заявления;

– согласование. Пользователь проверяет информацию, представленную в форме. Система выполняет автоматическую проверку корректности данных. После согласования заявление переходит на этап оценки, где оно проверяется ответственным лицом;

– оценка. Система формирует данные для оценки заявления, которые отправляются на сервер. На сервере происходит анализ данных, включая проверку на соответствие доступным ресурсам и правилам предоставления комнат;

– изменение статуса и отправка формы. После успешной проверки статус заявления обновляется, и информация сохраняется в базе данных. Пользователь видит обновленный статус заявления в интерфейсе;

– обработка ошибок. В случае неожиданного ответа или ошибки система завершает процесс, уведомляя пользователя.

На рисунке 28 представлена архитектура модуля.

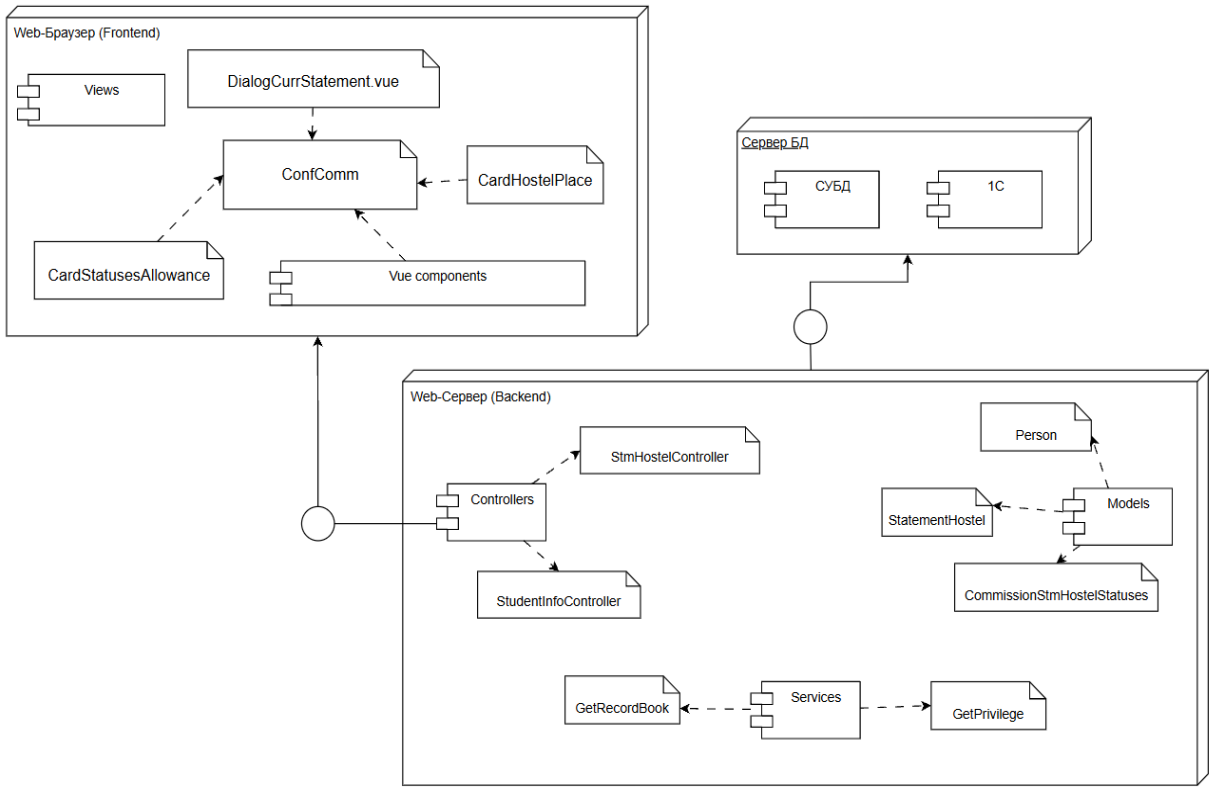

Рисунок 28 – Диаграмма компонентов модуля проверки заявлений

В основе всего поведения и зависимостей данного модуля лежит класс контроллера StatementHostel::Pichers::StmHostelController отвечающий за

управление процессами обработки заявлений на проживание в общежитиях университета. Контроллер использует layout 'application', требует аутентификацию пользователя и отключает проверку CSRF-токена для обработки внешних запросов. Внутри подключён модуль StatementHostel::StatementHostelApi, а для работы с системой 1С предусмотрен метод подключения conect_1c_api. Метод index возвращает список заявлений пользователя в формате JSON. Метод update управляет подтверждением или отклонением заявлений, а show собирает расширенные данные о поданных заявлениях с деталями о студенте, факультете и статусах. Вспомогательные методы process_eviction, process_relocation и create_distribution_person_department организуют работу с изменением статуса проживания студента. Контроллер активно взаимодействует с внешними SOAP-сервисами через Savon-клиент и возвращает все ответы в формате JSON для дальнейшей обработки на клиентской стороне.

3.1.4 Модуль распределения студентов по комнатам

Рассматриваемый модуль является одним из самых объемных в реализации, так как включает в себя множественный набор зависимостей и функций, необходимых для эффективного распределения студентов по комнатам общежитий. Его разработка потребовала тщательного проектирования архитектуры, способной обеспечить взаимодействие между клиентской и серверной частью, а также поддержку сложных процессов, таких как проверка доступности комнат, обработка заявлений и управление статусами. Особое внимание уделяется обработке данных в реальном времени, а также поддержке пользовательских операций через интуитивно понятный интерфейс.

Для детального понимания функционала данного модуля важно изучить ключевые этапы его работы, отражённые на рисунках 29 – 30.

Основные этапы работы модуля при его инициализации (рисунок 29):

– запрос и инициализация данных. При входе в личный кабинет происходит загрузка данных о доступных комнатах, закрепленных студентах и свободных местах. Взаимодействие интерфейса с сервером обеспечивает актуальность данных.

– фильтрация и обработка данных. После извлечения данных осуществляется их группировка и проверка на предмет несоответствий. Статистические показатели, такие как число свободных комнат, готовятся для отображения пользователю.

– выбор и закрепление комнаты. Пользователь выбирает комнату, а система выполняет проверку доступности. По результатам проверки данные отправляются на сервер для закрепления комнаты.

– управление ошибками и статусами. В случае возникновения ошибки или непредвиденного ответа процесс завершается с уведомлением пользователя. После успешного завершения операции система обновляет статус заявления.

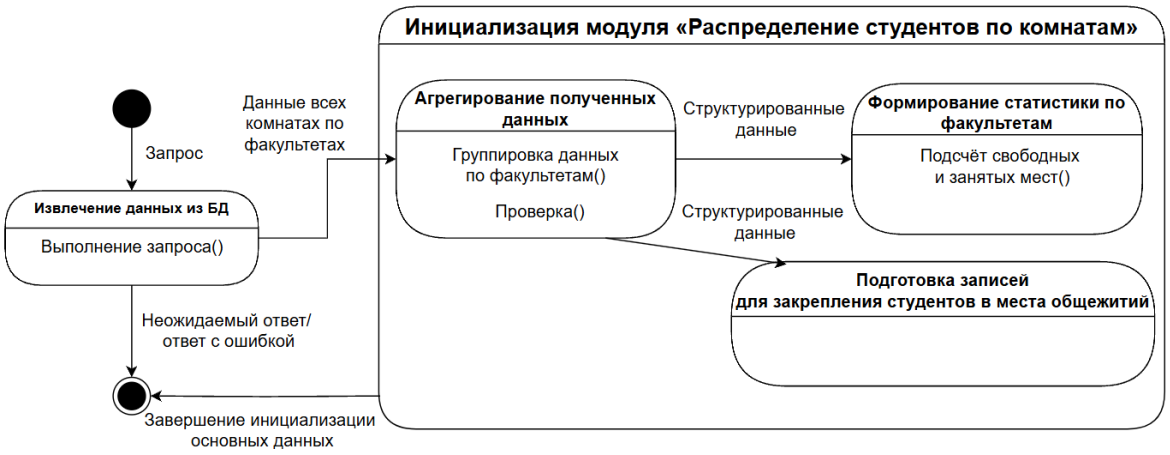

Рисунок 29 – Диаграмма состояний процесса инициализации модуля

Основные этапы работы модуля распределения студентов по местам (рисунок 30):

– инициализация данных. В начальном этапе происходит подсчёт рассмотренных и отклонённых заявлений. Выполняется вычисление различных статусов заявлений, что включает подсчёт статистических данных по каждой комнате. Дополнительно система отправляет SOAP-запросы для получения информации об абитуриентах, а полученные данные структурируются для дальнейшего использования.

– открытие формы для рассмотрения заявления. После инициализации отображается дополнительная информация, связанная с заявлением. Текущие

данные становятся основой для дальнейшего этапа работы с местами в общежитии.

– работа с данными о местах. Открывается окно информации о местах в общежитии. На этом этапе система подсчитывает количество свободных мест, выводит список заявлений, привязанных к текущему месту, а также проводит проверку корректности данных.

– закрепление студента за комнатой. Пользователь выбирает конкретную комнату. Далее происходит обновление статуса заявления, выполняется валидация и сохранение новых данных. Завершается процесс закрытием формы распределения.

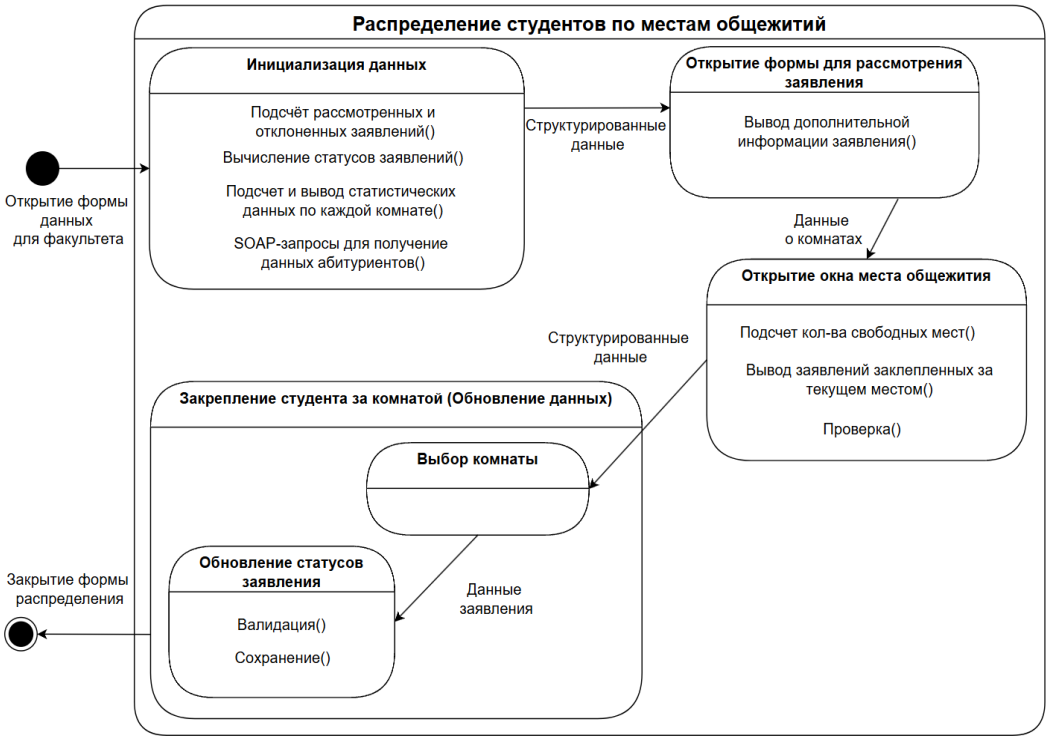

Рисунок 30 – Диаграмма состояний процесса распределения студентов

Структура модуля, описанная на рисунке 31, демонстрирует взаимодействие её ключевых компонентов, обеспечивая обработку заявлений и распределение студентов по местам в общежитии. Данная структура отражает сложную многослойную архитектуру веб-приложения, которая обеспечивает взаимодействие между пользователем и системой распределения студентов по местам в общежитии. Она состоит из трех основных уровней: фронтенд, бэкенд

и сервер базы данных, каждый из которых имеет ключевые компоненты, выполняющие свои роли.

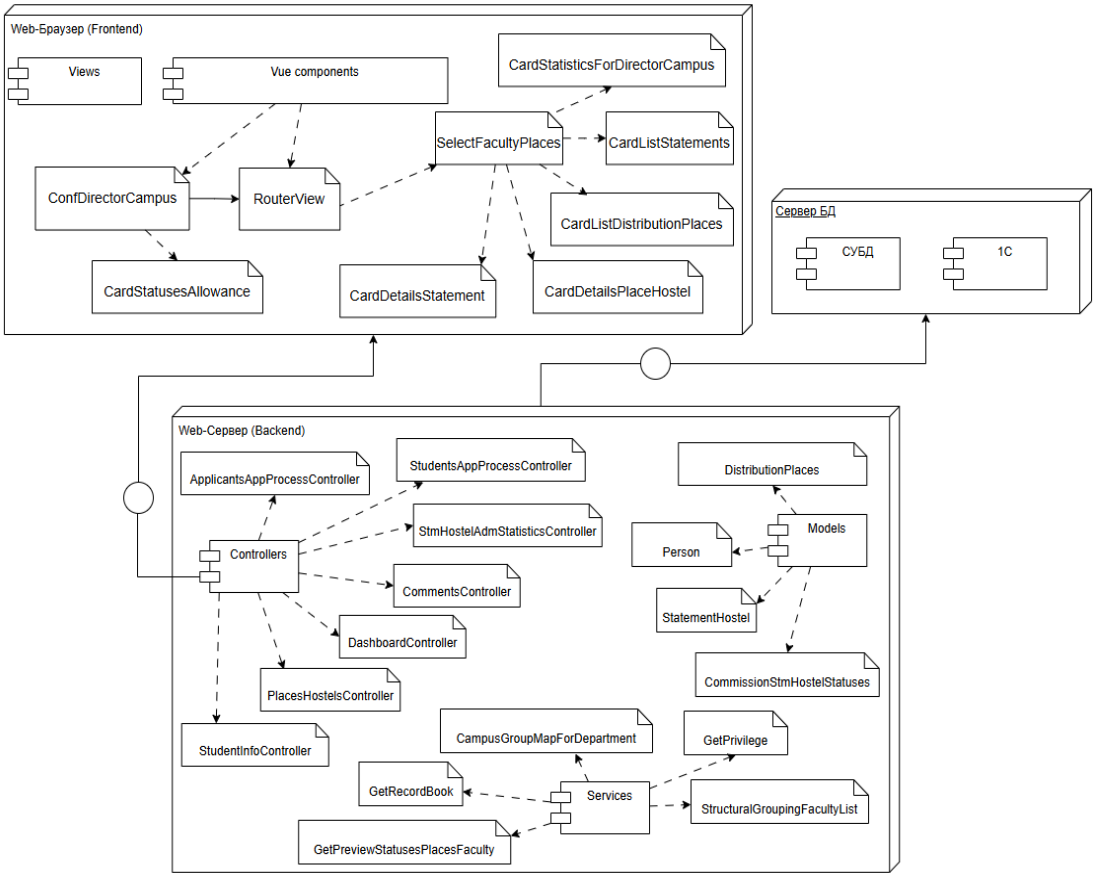

Рисунок 31 – Диаграмма компонентов модуля распределения студентов по комнатам

Фронтенд представлен множеством Vue-компонентов, которые обеспечивают интерактивность и удобство для пользователя. Например, компонент RouterView отвечает за маршрутизацию между различными экранами приложения, что позволяет пользователю легко перемещаться между функциями. Компоненты, такие как CardStatusesAllowance и CardDetailsStatement, играют роль визуализации информации о статусах заявлений и деталях заявлений соответственно. Это важно для оперативного предоставления студенту ключевых данных без необходимости обращения в администрацию. Также стоит отметить компонент SelectFacultyPlaces, который позволяет выбирать места, связанные с факультетом. Он критически важен для настройки распределения по комнатам с учетом специфики учебных подразделений. Более специализиро-

ванные компоненты, такие как CardStatisticsForDirectorCampus, предоставляют статистику для анализа, облегчая работу административного персонала. Такое разделение задач между компонентами обусловлено необходимостью обеспечения модульности и удобства добавления новых функций.

На серверной стороне архитектура включает набор контроллеров и сервисов, обеспечивающих обработку данных и выполнение бизнес-логики. Например, ApplicantsAppProcessController и StudentsAppProcessController обрабатывают запросы, связанные с заявками и студентами, что позволяет четко разграничивать роли пользователей системы. Это важно для сохранения целостности данных и управления правами доступа.

Сервисы, такие как StructuralGroupingFacultyList, выполняют задачи группировки факультетов. Они играют ключевую роль в оптимизации работы приложения, снижая нагрузку на сервер при выполнении сложных запросов. Модель StatementHostel организует хранение информации о заявлениях, что позволяет быстро извлекать данные и проводить их обработку. Такой подход обеспечивает надежность и согласованность при работе с данными.

Контроллеры, как DashboardController, обрабатывают запросы для формирования главного экрана, включая визуализацию статистики. Это упрощает взаимодействие с пользователем, предоставляя важную информацию в удобной форме.

Таблица 3 содержит обобщённую информацию о ключевых характеристиках контроллера «StatementHostel::Commissions::StudentsAppProcessController» , отвечающего за обработку заявлений студентов в процессе их рассмотрения комиссией.

Таблица 3 – Описание класса «StatementHostel::Commissions::StudentsAppProcessController»

Элемент	Описание
1	2
Класс	StatementHostel::Commissions::StudentsAppProcessController – контроллер для обработки заявлений студентов в процессе их рассмотрения комиссией и предоставления связанной информации.

Продолжение таблицы 3

Наследование	Наследуется от базового класса ApplicationController.
Layout	Используется общий макет 'application'.
Фильтры	before_action :authenticate_user! – проверка аутентификации пользователя перед выполнением всех действий. skip_before_action :verify_authenticity_token – отключение проверки CSRF-токена для всех методов.
Права доступа	Доступ к действиям открыт только для аутентифицированных пользователей. Дополнительные проверки прав (например, роли члена комиссии) в коде контроллера отсутствуют.
Методы контроллера	index – получение списка заявлений студентов для указанного факультета через сервис StmHostelServices::GetStmHostelsForDepartment, по department_id. show – получение подробной информации о конкретном заявлении, включая данные студента и статусы рассмотрения комиссией. get_details_place – получение распределения студентов по местам общежитий для указанного факультета (place_id), с детальной информацией о студенте и приказе на поселение.
Приватные методы	Отсутствуют в этом контроллере. Все методы публичные.
Зависимости	Сервис: StmHostelServices::GetStmHostelsForDepartment. Модели: StatementHostel, DistributionPersonDepartment, Person, Department, HostelOrder, CommissionStmHostelStatus.

3.1.5 Модуль проверки договора на проживание

В рамках описания работы модуля оформления договоров на проживание в общежитии ниже представлена текстовая расшифровка и иллюстрация бизнес-процесса. На рисунке 32 схематично отображён процесс проверки и согласования договора, а также его ключевые этапы – от извлечения данных до финального подписания. Текстовое описание дополняет диаграмму, подробно поясняя последовательность действий, взаимодействие между компонентами системы и обработку возможных исключительных ситуаций. Такой подход

позволяет комплексно оценить надёжность и логику реализации процесса в системе.

Основной процесс состоит из нескольких ключевых частей. Сначала система инициирует запрос на получение данных из базы. Это позволяет извлечь необходимую информацию о заявлении и договоре. Затем данные проходят стадию подготовки и обработки, где формируется запись с подробными сведениями о заявлении и договоре. После этого наступает этап согласования, в рамках которого договору присваивается уникальный номер и осуществляется подписание через сторонний сервис с использованием национальной электронной подписи.

В финальной части проверки предусмотрены механизмы обработки непредвиденных ситуаций, включая ошибки или неожиданные ответы. Такая структура построена для обеспечения надежности и последовательности выполнения операций, что минимизирует риски сбоев на любом этапе процесса.

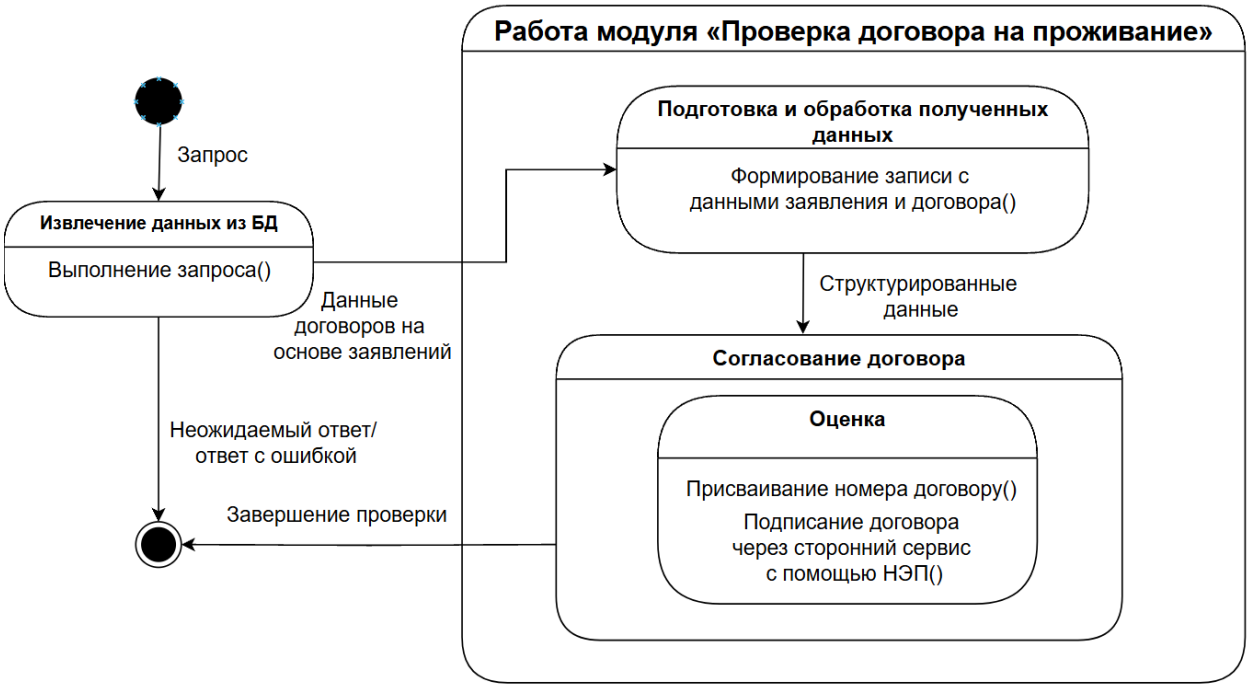

Рисунок 32 – Диаграмма состояний процесса проверки договора на проживание

Диаграмма компонентов, изображенная на рисунке 33, отражает процесс обработки данных, их проверки и согласования, включая этапы формирования

электронной подписи, фиксации статусов документов и их автоматической передачи в интегрированные системы.

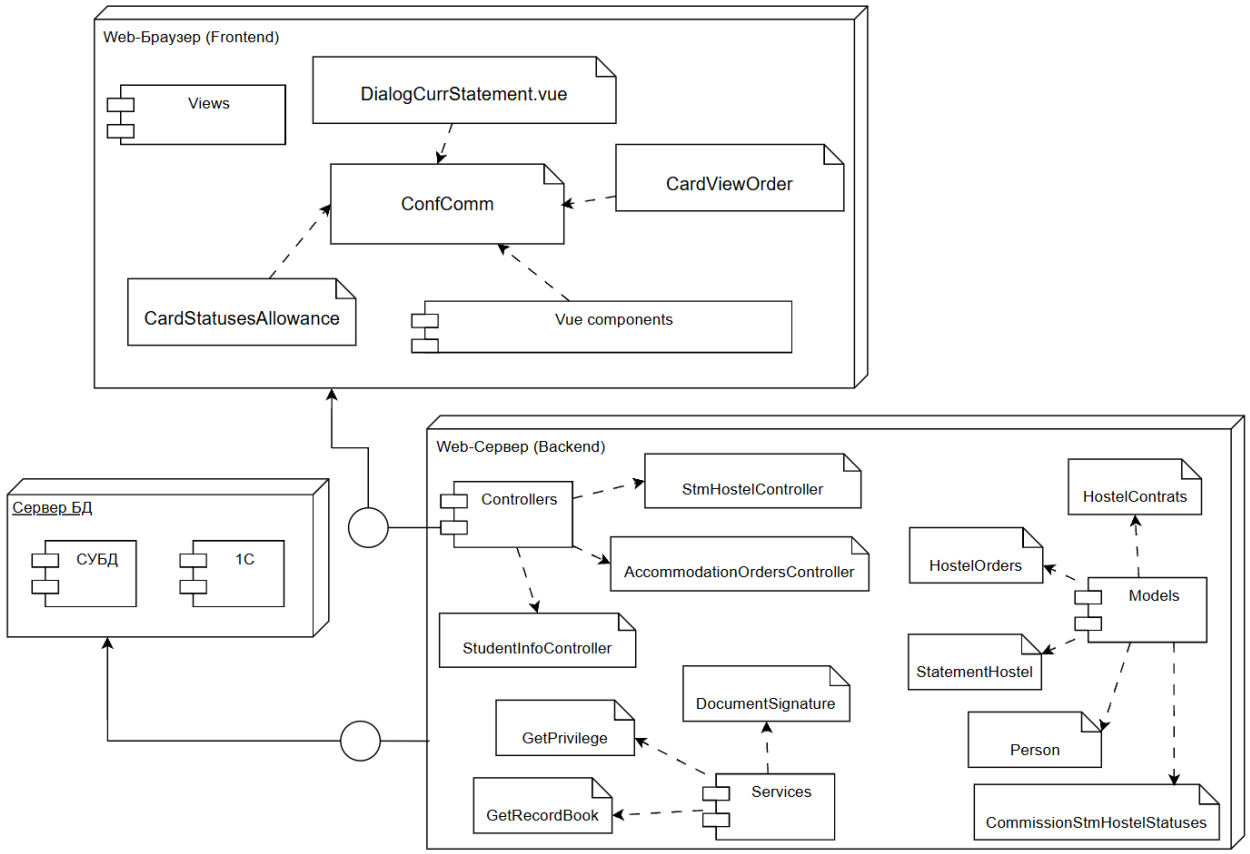

Рисунок 33 – Диаграмма компонентов модуля проверки договора на проживание

Таблица 4 описывает функциональные особенности контроллера «AccommodationOrdersController», который отвечает за управление приказами о заселении студентов в общежития.

Таблица 4 – Описание контроллера «AccommodationOrdersController»

Элемент	Описание
Класс	StatementHostel::Commissions::AccommodationOrdersController – отвечает за работу с приказами о заселении студентов в общежитие.
Наследование	Наследует от ApplicationController.
Layout	Используется общий макет 'application'.
Фильтры	before_action :authenticate_user! – проверка авторизации пользователя. before_action :conect_1c_api, only: [:create] – подключение к API 1С при создании приказа.

Продолжение таблицы 4

Права доступа	Доступ ограничен авторизованными пользователями. Создание приказа (create) разрешено только пользователям с ролью dean_ex_work.
Методы контроллера	index – получение списка всех приказов на основе политики доступа (StmHostelPolicy::Scope). Возвращает данные в формате JSON. create – создание приказа о заселении. Формируется запрос в 1С через SOAP-клиент Savon. При успешной обработке создаётся локальная запись о приказе
Приватные методы	order_complete – вспомогательная функция для оформления элементов приказа: преобразует данные о проживающих в человекочитаемый вид. conect_1c_api – инициализация подключения к API 1С с помощью клиента Savon на основе настроек из config/settings.yml.
Зависимости	Зависит от внешней системы 1С (API создания приказов через SOAP-запросы). Использует модели Person, FacultiesDorm, HostelOrder.

Планируется расширение функциональности контроллера для интеграции механизма электронной подписи договоров о заселении студентов в общежитие с использованием сервиса Контур.Сайн через собственный сервис DocumentSignature.

Процесс будет организован следующим образом: после создания записи о приказе на заселение в методе create, автоматически будет генерироваться текст договора на основе информации о студенте, выбранном месте проживания и сроках проживания. Для этого будет применяться заранее подготовленный шаблон, в который динамически подставляются актуальные данные.

Далее договору будет присваиваться уникальный регистрационный номер, например в формате ПР-ОБЩ-{год}-{идентификатор}, чтобы обеспечить отслеживаемость всех заключённых соглашений.

Инициализация подписания будет происходить через сервис DocumentSignature, который выполняет:

– загрузку сгенерированного договора;

– постановку подписи сначала членом комиссии, ответственным за размещение студентов.

После успешной подписи комиссией документ автоматически будет отправляться студенту для последующей подписи. Подписание осуществляется через API Контур.Сайн, что позволяет работать как через прямые ссылки, так и через уведомления по электронной почте.

Процесс также будет включать отслеживание статуса подписания договора: в системе будет учитываться, находится ли документ на стадии ожидания подписи, подписан обеими сторонами или возникли ошибки или отклонение. При успешном подписании обеими сторонами договор будет сохраняться в системе и будет доступен для скачивания и печати через личный кабинет пользователя.

В случае ошибок на любом этапе взаимодействия с Контур.Сайн будет предусмотрено:

– автоматическое логирование ошибок;

– отображение пользователю понятных уведомлений о необходимости повторить операцию или обратиться за поддержкой;

– откат неудачных транзакций для сохранения целостности данных.

3.2. Примеры фактического тестирования программного продукта

В процессе тестирования программного продукта особое внимание уделялось проверке качества реализации пользовательского интерфейса. Это обусловлено тем, что интерфейс является основной точкой взаимодействия пользователя с системой, и от его характеристик напрямую зависит удобство работы с приложением. Ключевыми требованиями являлись: наглядность, понятность, информативность и интерактивность.

Модуль «Распределение комнат по факультетам». Данный модуль берет на себя обязанность в подготовки данных о комнатах общежитий, а также их распределении между факультетами. При открытии страницы отображается краткое сведения о местах общежитий и непосредственно распределенных местах. На рисунке 34 изображена статистика по общежитиям, сколько занято и

свободно мест, что дает увидеть пользователю текущее состояние используемых жилых ресурсов.

Рисунок 34 – Список общежитий с информацией о местах

Далее на рисунке 35 отображается базовая сгруппированная информация о уже закрепленных за определенным факультетам комнат с дополнительной настройкой числа предоставляемых комнат. На ряду с этим каждая карточка является интерактивной, нажав на которую можно увидеть расширенную информацию.

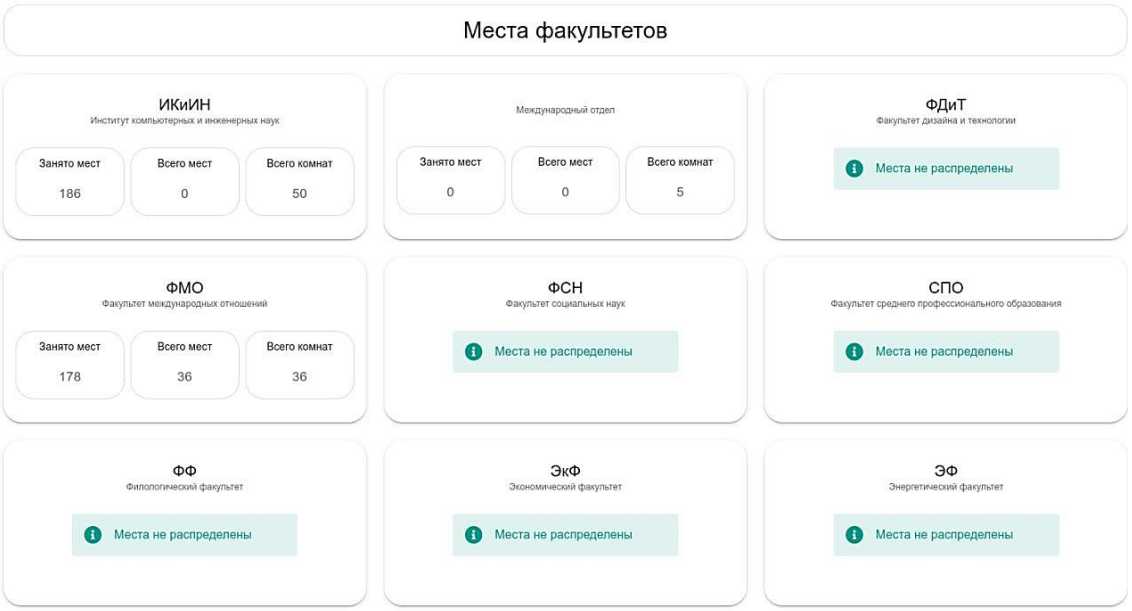

Рисунок 35 – Факультетов с информацией о местах и комнатах

Диалоговое окно при открытии карточки изображено на рисунке 36. Оно включает в себя поиск комнат, группировку и их распределение, для наглядности вся информация поддерживается цветными статусами.

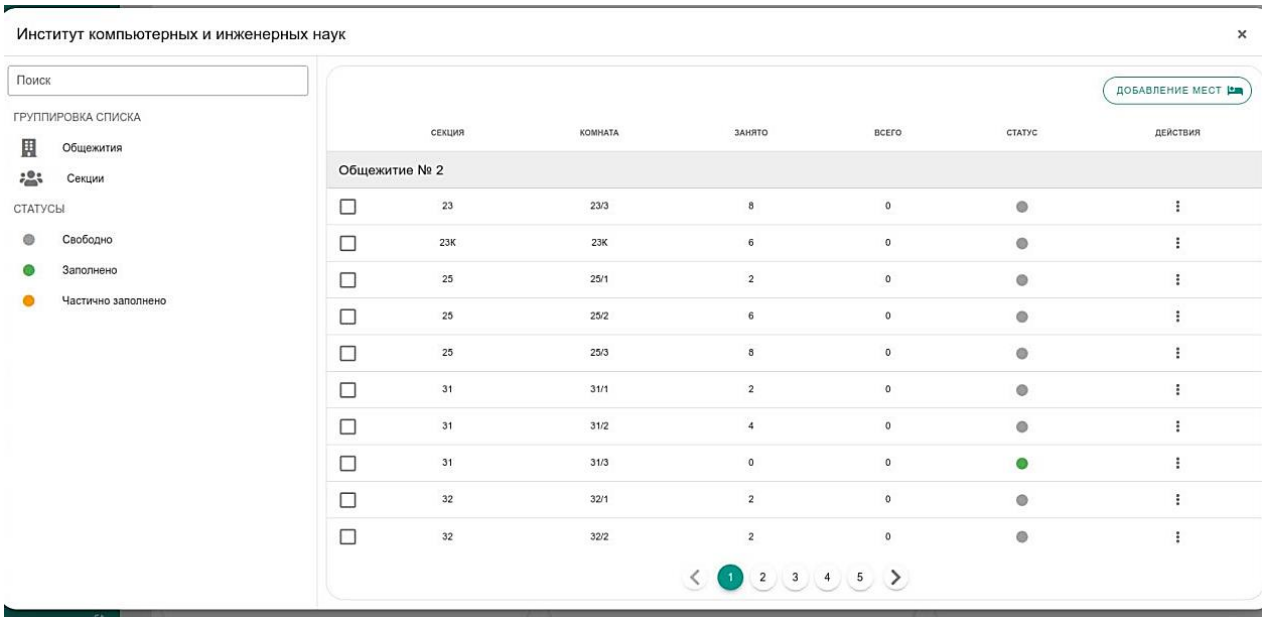

Рисунок 36 – Диалоговое окно для распределения комнат в факультет

Основной частью функционала является действие добавление мест, изображено на рисунке 37. Наглядное отображение мест, возможность добавления нескольких позиций, а также учет занятости строит положительный опыт пользователя.

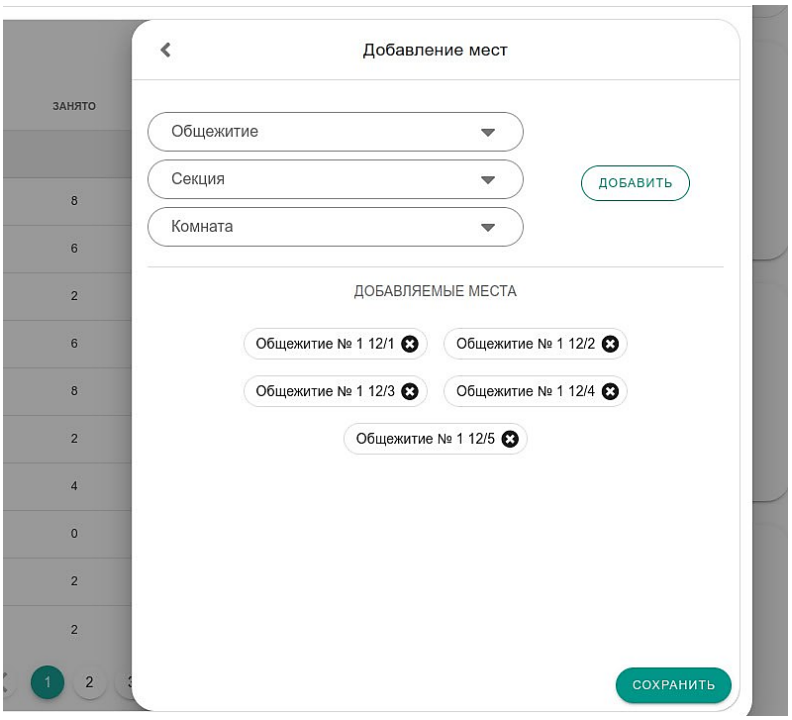

Рисунок 37 – Функциональное меню для распределения комнат

Модуль «Подача заявлений». Перед подачей заявления на заселения, студент может видеть карточку (Рисунок 38) персональной информации, содержащей рейтинг академической успеваемости, факультет, уровень образования и т.д. Рейтинг является главным основанием при отборе кандидатов на заселение, особенно при спорных моментах, также на ряду с этим существует возможность его подробного просмотра.

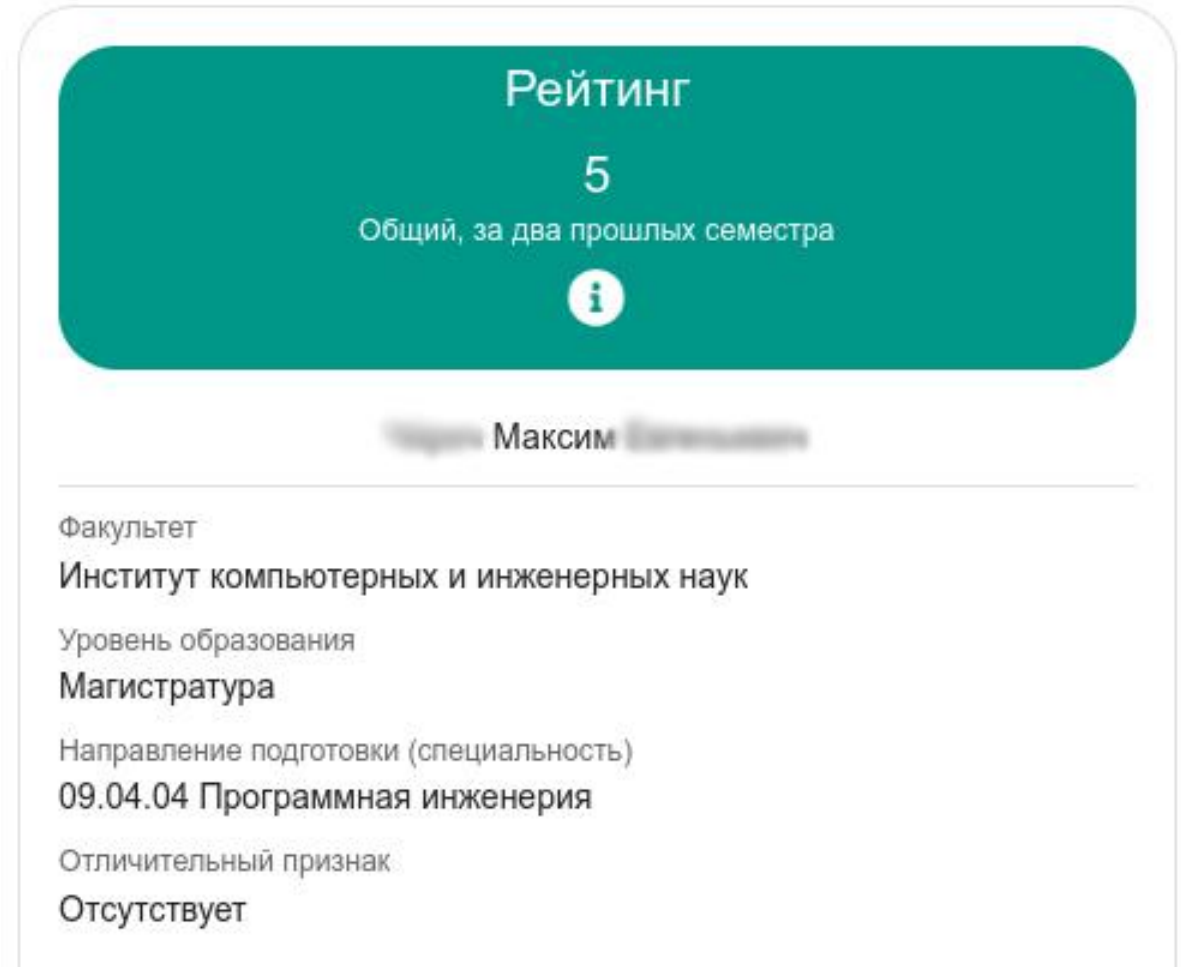

Рисунок 38 – Вид карточки персональной информации

На первом этапе студент может понять проходит ли он по условиям подачи заявления. Важными факторами для данного процесса является наличие иногородней прописки и актуальной пройденной флюорографии, ниже отображается последнее согласованное заявления, показывающее состояние договора на общежитие и соответствующие ему состояние приказа (Рисунок 39).

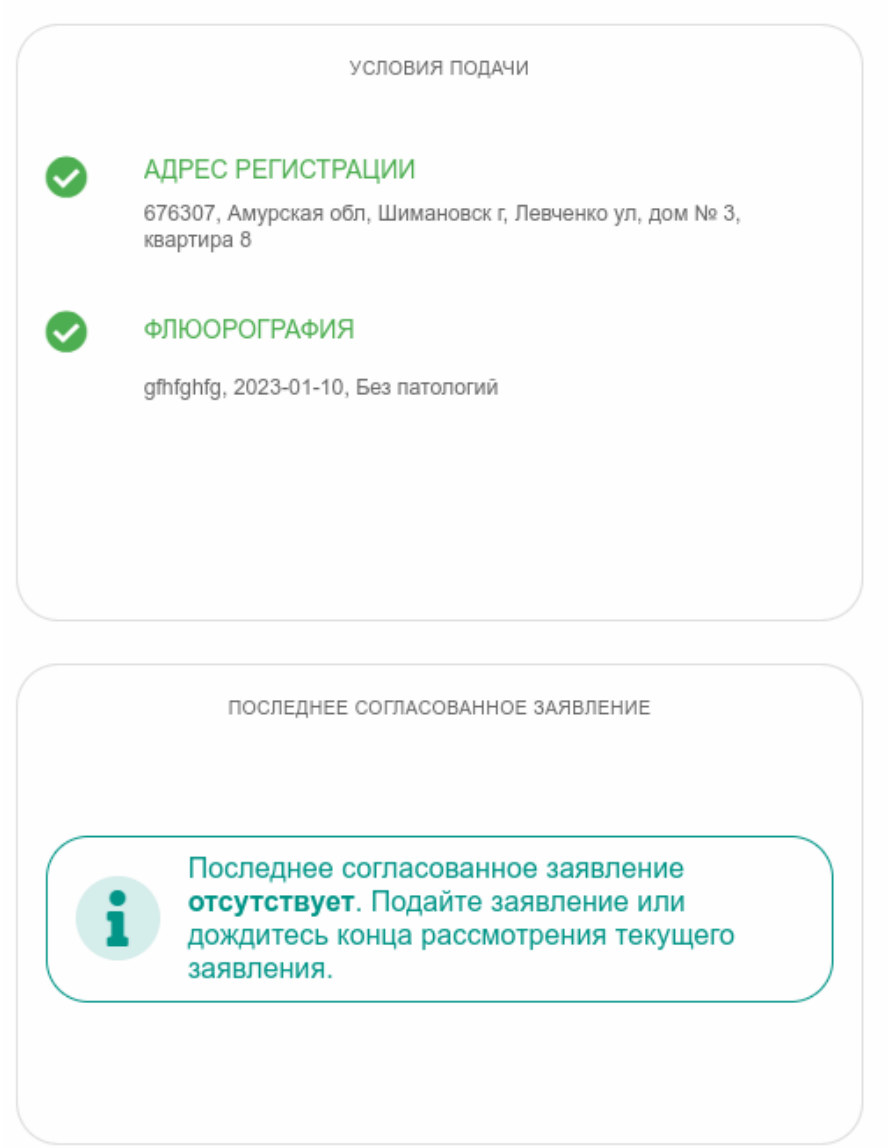

Рисунок 39 – Вид отображения условий подачи и последнего одобренного заявления

В зависимости от состояния последнего согласованного заявления вид карточки меняется. Она имеет как базовую информацию заявления, так и возможность просмотра ранее формированного договора на проживание. Альтернативный вид карточки изображен на рисунке 40.

После авторизации пользователь может приступить к оформлению заявления. При нажатии на кнопку «Подать заявление» открывается всплывающее окно, в котором пошагово отображён весь процесс подачи. Первый этап содержит информационный блок: здесь представлено краткое описание работы системы, а также основные правила подачи заявлений (см. рисунок 41). Текущий

шаг помогает пользователю заранее ознакомиться с порядком действий и требованиями.

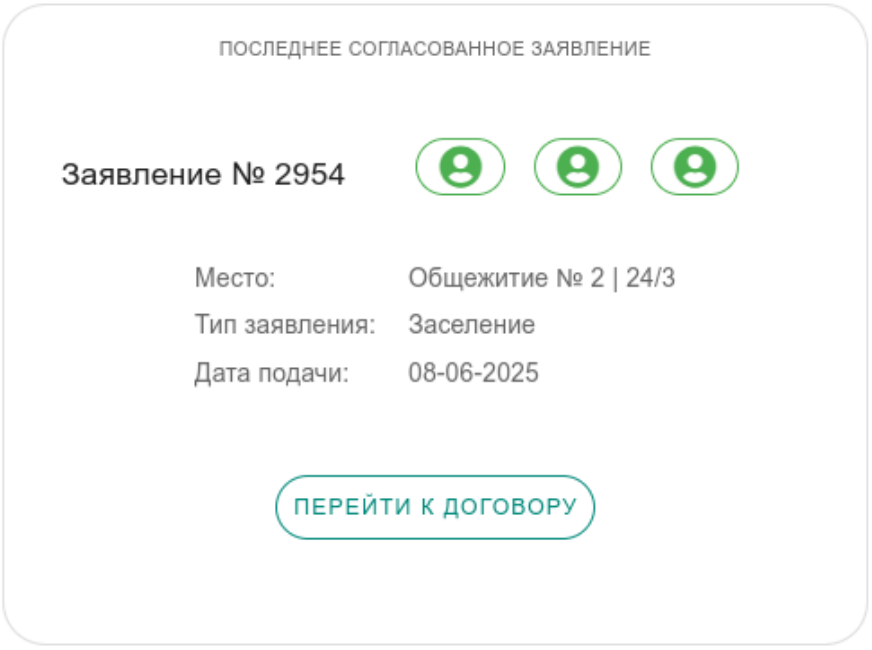

Рисунок 40 – Альтернативный вид карточки «Последнее согласованное заявление»

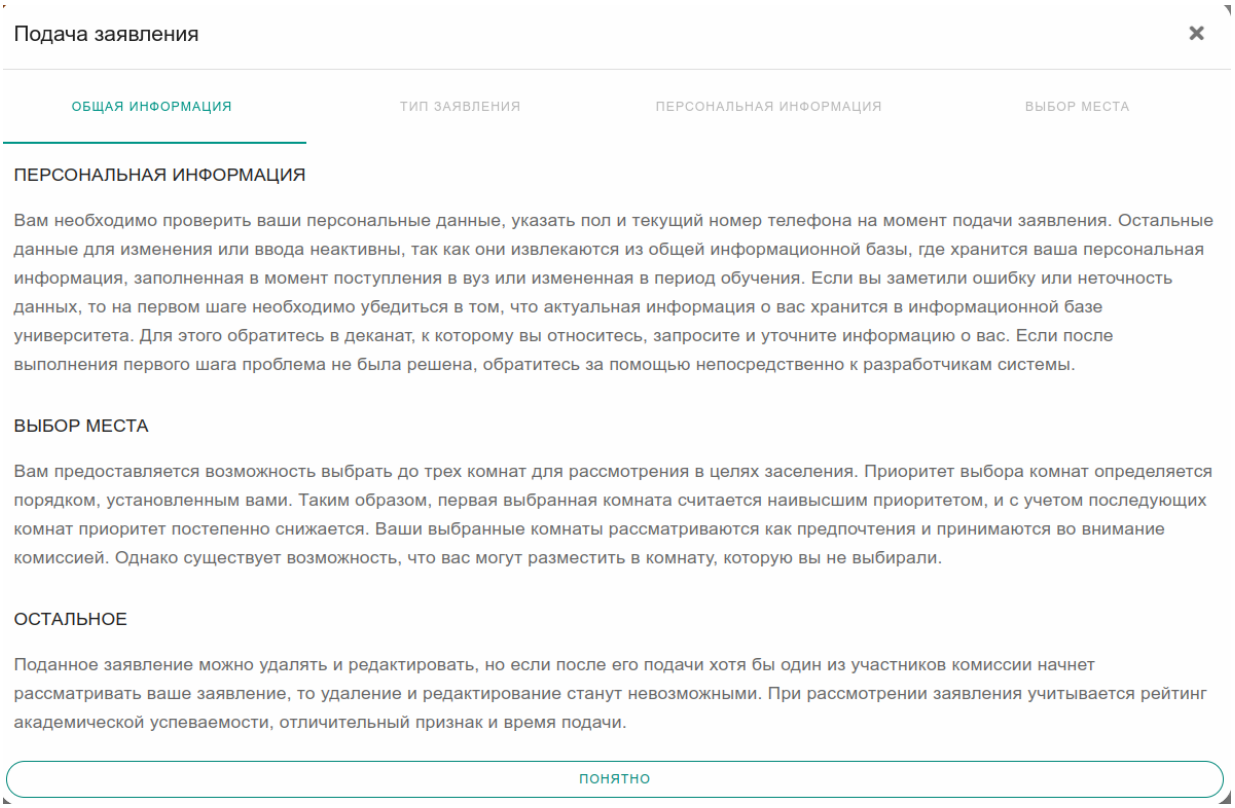

Рисунок 41 – Первый этап при подаче заявления

На втором этапе пользователь переходит к выбору типа заявления. Си-

стема автоматически анализирует доступные данные и на их основе ограничивает список доступных вариантов, предлагая только те, которые соответствуют текущему статусу пользователя (см. рисунок 42).

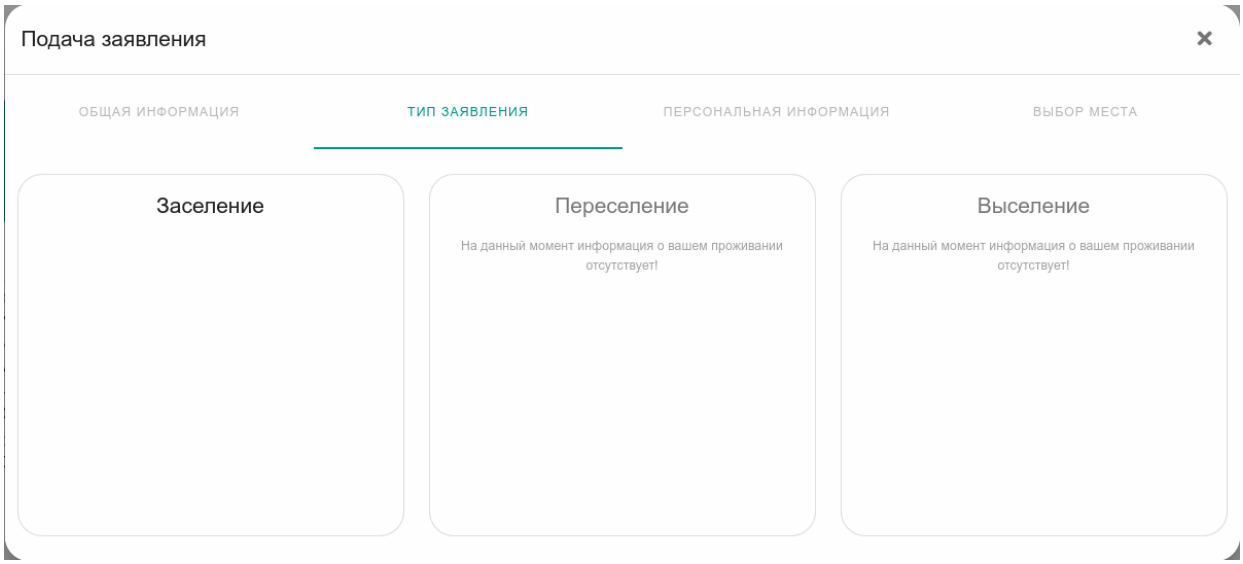

Рисунок 42 – Второй этап подачи заявления

На третьем этапе пользователь проверяет персональные данные и дополняет их необходимой информацией – указывает пол и актуальный номер телефона (см. рисунок 43). Большинство полей уже автоматически заполнены, так как информация подгружается из общей базы данных университета, что приводит к сокращению времени заполнения и снижению вероятность ошибок со стороны пользователя.

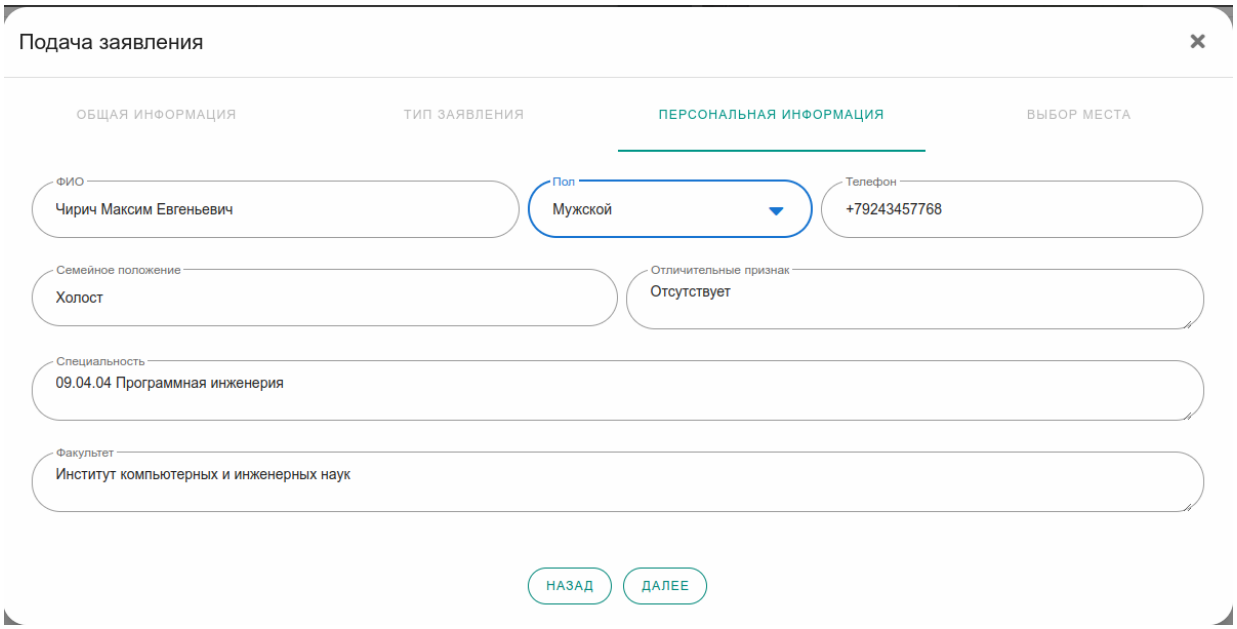

Рисунок 43 – Третий этап при подаче заявления

На следующем этапе пользователь выбирает до трёх желаемых мест в общежитии, расставляя их по степени приоритета – первой указывается комната с наивысшим предпочтением. С каждой добавленной позицией приоритет автоматически понижается.

Список доступных мест формируется на основе факультета, к которому относится пользователь, и отображается факультетно (рисунок 44). После выбора трёх комнат система уведомляет о том, что лимит заполнен. При попытке отправки заявления система проводит валидацию: поля с выбором мест не должны быть пустыми. Только после успешной проверки становится доступна кнопка «Подать».

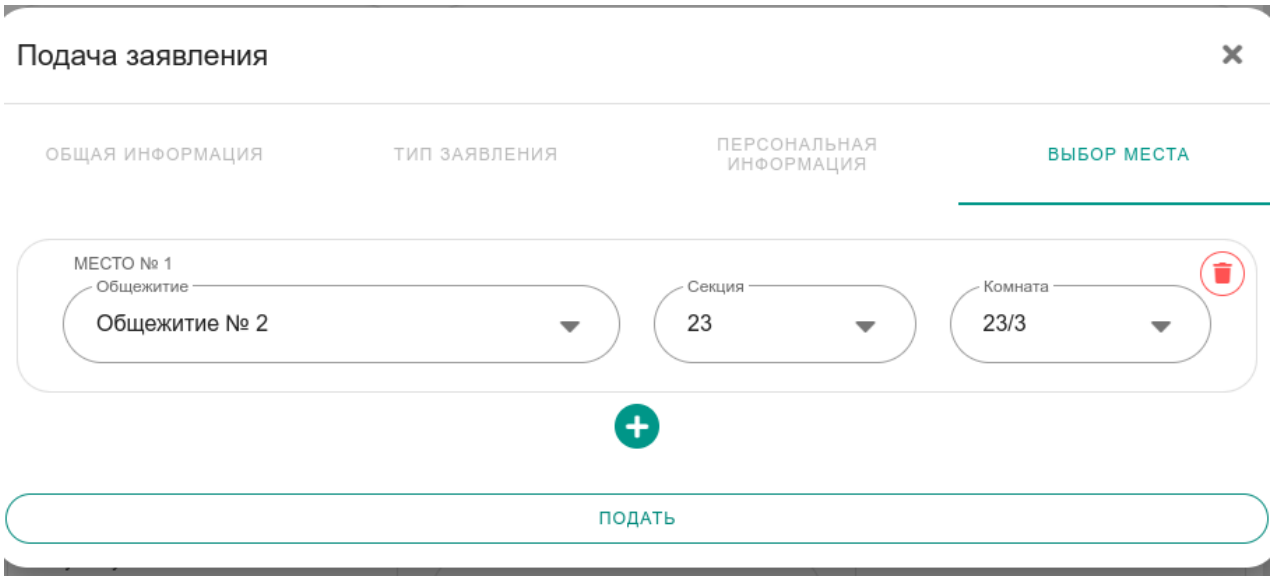

Рисунок 44 – Последний этап при подаче заявления

После успешной подачи заявление отображается на главной странице с краткой информацией (см. рисунок 45). Пользователь получает возможность просмотреть его, внести изменения или удалить при необходимости. Также доступен просмотр расширенной информации, включая оценки от членов комиссии (рисунок 44). Однако, как только хотя бы один участник комиссии приступит к рассмотрению заявления, редактирование и удаление станут недоступны. Данный функционал обеспечивает сохранность данных и исключает вмешательство в процессе проверки.

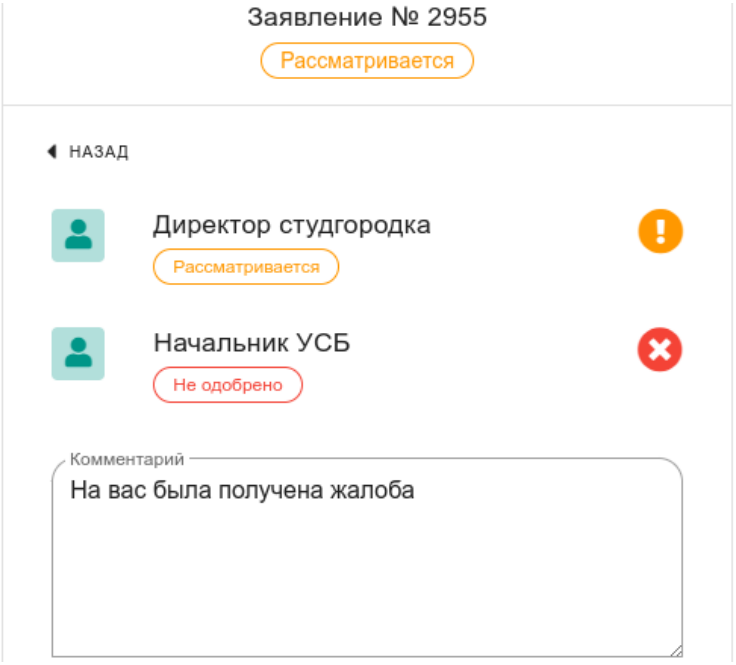

Рисунок 44 – Подробный вид оценки участника комиссии

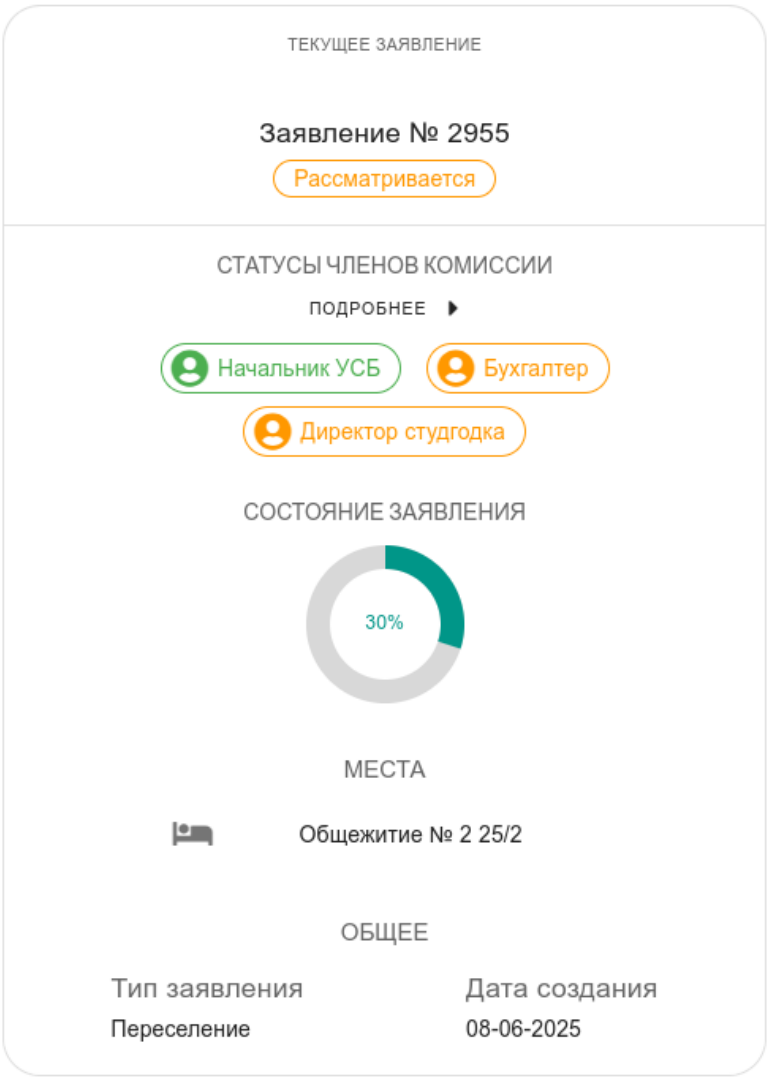

Рисунок 45 – Карточка поданного заявления

Модуль «Распределение студентов по комнатам». Процесс закрепления студента за конкретным местом в общежитии проходит следующим образом: на начальном этапе осуществляется инициализация данных, в рамках которой подсчитываются рассмотренные и отклонённые заявления, вычисляются различные статусы и формируется статистика по каждой комнате, данная работа модуля представляется на рисунке 45. Также выполняются SOAP-запросы к внешним системам для получения информации об абитуриентах, после чего данные структурируются для последующей обработки. Далее открывается форма для рассмотрения конкретного заявления, где отображается расширенная информация о студенте и его предпочтениях.

Следующим этапом становится работа с доступными местами: система подсчитывает количество свободных комнат, показывает список заявлений, уже связанных с выбранными местами, и проверяет корректность всех данных. При выборе пользователем определённой комнаты система обновляет статус заявления, проводит валидацию выбранного места и сохраняет новую информацию в базе данных. Завершается процесс закрытием формы распределения.

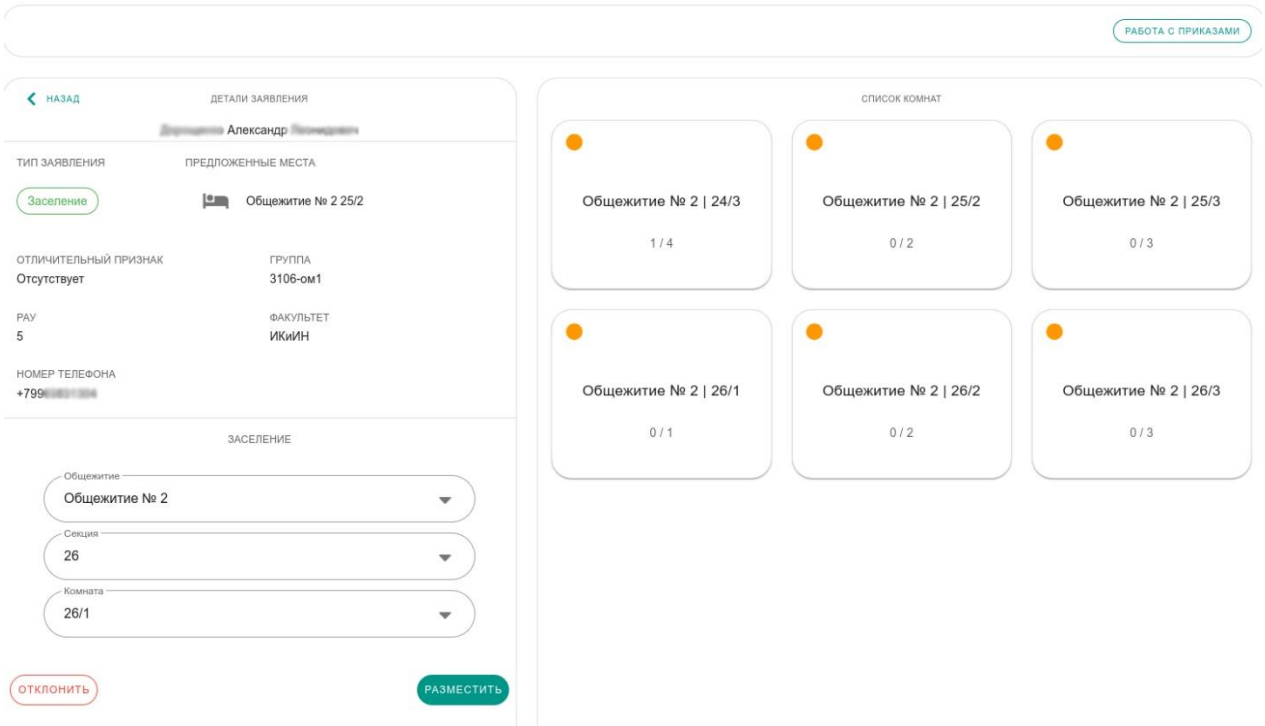

Рисунок 46 – Общий вид системы при распределении студентов по комнатам

Общее описание процесса работы приложения: после подачи студентом

электронного заявления оно поступает в общий реестр на согласование. Процедура согласования предполагает последовательную экспертизу несколькими участниками комиссии. Первоначально ответственное лицо осуществляет проверку корректности представленных данных о студенте и соответствия заявки установленным требованиям. На следующем этапе бухгалтерская служба верифицирует финансовые аспекты, а именно наличие подтвержденной оплаты проживания либо документально обоснованных прав на льготное заселение. Затем начальник Управления безопасности (УСБ) проводит оценку на предмет соответствия кандидата требованиям внутреннего распорядка и безопасности проживания. Каждое действие участников комиссии сопровождается обязательным обновлением статуса заявления в единой информационной системе, что создает условия для прозрачного мониторинга хода согласования в режиме реального времени. После успешного прохождения всех проверочных этапов автоматически формируется проект приказа о заселении, подлежащий утверждению комиссией в установленном порядке.

Ключевым элементом оптимизации финальной стадии процесса стала интеграция университетской системы с сервисом квалифицированной электронной подписи (КЭП) «Контур.Сайн». Данная доработка информационной системы преследует цель кардинального повышения эффективности документооборота. Утвержденный приказ о заселении подписывается электронной подписью уполномоченного члена комиссии непосредственно в системе. Далее сформированный на его основе договор найма жилого помещения в общежитии направляется студенту для подписания его собственной электронной подписью через интегрированный сервис. Использование «Контур.Сайн» в качестве внешнего, но надежно интегрированного компонента системы обеспечивает несколько значимых преимуществ. Во-первых, достигается существенное сокращение сроков процедуры заселения за счет исключения необходимости личного присутствия студента для подписания бумажных документов. Во-вторых, обеспечивается полная юридическая значимость электронных доку-

ментов благодаря применению квалифицированной электронной подписи, соответствующей требованиям законодательства. В-третьих, минимизируются риски потери документов и ошибок ручного ввода данных. В-четвертых, студент получает мгновенный доступ к подписанному электронному экземпляру договора для самостоятельной печати или дальнейшего электронного использования, что окончательно формализует процедуру заселения. Таким образом, внедрение взаимодействия с «Контур.Сайн» представляет собой целенаправленную доработку системы, направленную на достижение безбумажного, юридически безупречного и максимально удобного для всех участников завершения процесса заселения в студенческое общежитие.

ЗАКЛЮЧЕНИЕ

В рамках данной магистерской диссертации была разработана и внедрена информационная система автоматизации процесса заселения студентов в общежития Амурского государственного университета. Работа охватила весь цикл разработки программного обеспечения – от анализа требований и проектирования архитектуры до реализации, интеграции и тестирования системы в действующей цифровой инфраструктуре вуза.

Разработанный программный модуль позволяет значительно повысить прозрачность, управляемость и эффективность процесса заселения, снизить административную нагрузку на сотрудников, а также улучшить взаимодействие студентов с университетскими сервисами. Интеграция с личным кабинетом на сайте АмГУ и системой 1С:Университет обеспечивает сквозной документооборот, своевременное обновление данных и поддержку актуального состояния всех связанных процессов.

В ходе реализации были применены современные технологии веб-разработки, такие как Ruby on Rails и Vue.js, что позволило создать масштабируемую, модульную и удобную в сопровождении систему. Также были внедрены инструменты визуализации структуры комнат и формирования статистических отчётов, что расширяет функциональные возможности системы и повышает её практическую значимость.

Таким образом, цели, поставленные в начале работы, были достигнуты. Результаты исследования и практической реализации могут быть использованы в дальнейшем для расширения функциональности системы, автоматизации дополнительных процессов, а также внедрения аналогичных решений в других образовательных организациях.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Анализ требований безопасности информационных систем [Электронный ресурс]. – Режим доступа: https : //cyberleninka.ru/article/n/analiz-trebovaniy-bezopasnosti-informatsionnyh-sistem. – 02.12.2024.

2 Автоматизированные информационные системы управления учебным процессом вуза: практическое исследование [Электронный ресурс]. – Режим доступа : https://cyberleninka.ru/article/n/avtomatizirovannye-informatsionnye-sistemy-upravleniya-uchebnym-protsessom-prakticheskoe-issledovanie. – 02.12.2024.

3 Основы UML. Кому и зачем он нужен [Электронный ресурс]. – Режим доступа : https://systems.education/who-uses-uml. – 01.12.2024.

4 Направления автоматизации системы управления вузом [Электронный ресурс]. – Режим доступа : https://cyberleninka.ru/article/n/napravleniya-avtomatizatsii-sistemy-upravleniya-vuzom/viewer. – 29.06.2024.

5 Оптимизация построения информационной системы управления вузом: концептуальные подходы [Электронный ресурс]. – Режим доступа : https://cyberleninka.ru/article/n/optimizatsiya-postroeniya-informatsionnoy-sistemy-upravleniya-vuzom-kontseptualnye-podhody. – 03.12.2024.

6 Прогрессивный JavaScript-фреймворк [Электронный ресурс]. – Режим доступа : https://ru.vuejs.org/. – 15.12.2024.

7 Проектирование информационных систем [Электронный ресурс]. – Режим доступа : https://elar.urfu.ru/bitstream/10995/. – 03.12.2024.

8 Разработка мобильных и веб-приложений [Электронный ресурс]. – Режим доступа : https://smartum.pro/ru/blog-ru/razrabotka-mobilnykh-i-web-prilozheniy/. – 15.12.2024.

9 Распределённый алгоритм системного взаимного информационного согласования в многокомплексных вычислительных системах [Электронный ресурс]. – Режим доступа : https://cyberleninka.ru/article/n/raspredelyonnyy-algoritm-sistemnogo-vzaimnogo-informatsionnogo-soglasovaniya-v-

mnogokompleksnyh-vychislitelnyh-sistemah. – 03.12.2024.

10 Ruby on Rails по-русски [Электронный ресурс]. – Режим доступа : http://rusrails.ru. – 16.03.2023.

11 Компонентная диаграмма – Унифицированный язык моделирования (UML) [Электронный ресурс]. – Режим доступа : https://www.geeksforgeeks.org/component-based-diagram/. – 11.12.2024.

12 Модульная архитектура: что, как и почему? [Электронный ресурс]. – Режим доступа : https://proglib.io/p/modulnaya-arhitektura-chto-kak-i-pochemu. – 01.12.2024.

13 Фултон, Х. Программирование на языке Ruby / Х. Фултон ; под ред. Д. А. Мовчан ; пер. А. А. Слинкин. – 2-е изд. // Саратов : Профобразование. – 2019. – 685 с.

14 Хартл, М. Ruby on Rails для начинающих / М. Хартл // Москва: ДМК Пресс. – 2017. – 572 с.

15 A Gentle Introduction to UML Class Diagrams [Электронный ресурс]. – Режим доступа : https://www.cs.utah.edu/~germain/PPS/Topics/umlClassDiagrams.pdf. – 15.12.2024.

16 Agile Modeling and the UML [Электронный ресурс]. – Режим доступа : https://agilemodeling.com/essays/agileModelingUML.htm. – 15.12.2024.

17 Component Diagram Tutorial [Электронный ресурс]. – Режим доступа : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/. – 15.12.2024.

18 Introduction to UML 2.5 [Электронный ресурс]. – Режим доступа : https://www.omg.org/spec/UML/2.5/. – 15.12.2024.

19 Introduction to Vue.js [Электронный ресурс]. – Режим доступа : https://vuejs.org/guide/introduction.html. – 15.12.2024.

20 JavaScript frameworks comparison [Электронный ресурс]. – Режим доступа : https://www.freecodecamp.org/news/javascript-frameworks-explained/. – 15.12.2024.

21 Model-View-Controller (MVC) Explained [Электронный ресурс]. – Режим доступа : https://developer.mozilla.org/en-US/docs/Glossary/MVC. – 15.12.2024.

22 Ruby Language Documentation [Электронный ресурс]. – Режим доступа : https://www.ruby-lang.org/en/documentation/. – 15.12.2024.

23 Smartum: Блог разработчиков [Электронный ресурс]. – Режим доступа : https://smartum.pro/ru/blog-ru/. – 15.12.2024.

24 Software Architecture Patterns [Электронный ресурс]. – Режим доступа : https://martinfowler.com/articles/enterpriseArchitecture.html. – 15.12.2024.

25 Software Design Basics [Электронный ресурс]. – Режим доступа : https://www.geeksforgeeks.org/software-engineering-software-design-basics/. – 15.12.2024.

26 UML 2 Diagrams Overview [Электронный ресурс]. – Режим доступа : https://www.uml-diagrams.org/uml-2-diagrams.html. – 15.12.2024.

27 UML Component Diagrams Explained [Электронный ресурс]. – Режим доступа : https://www.lucidchart.com/pages/uml-component-diagram. – 15.12.2024.

28 UML Use Case Diagrams [Электронный ресурс]. – Режим доступа : https://www.uml-diagrams.org/use-case-diagrams.html. – 15.12.2024.

29 Using UML for Modeling Complex Systems [Электронный ресурс]. – Режим доступа : https://ieeexplore.ieee.org/document/12345678. – 15.12.2024.

30 What is UML? [Электронный ресурс]. – Режим доступа : https://sparxsystems.com/resources/uml2_tutorial/index.html. – 15.12.2024.