# Содержание

[Введение 7](#_Toc89976131)

[1 Описание предметной области 8](#_Toc89976132)

[2 Постановка задачи и обзор методов её решения 13](#_Toc89976133)

[2.1 Постановка задачи 13](#_Toc89976134)

[2.2 Описание методов решения поставленной задачи 14](#_Toc89976135)

[3 Функциональное моделирование на основе стандарта IDEF0 17](#_Toc89976136)

[4 Информационная модель системы и её описание 23](#_Toc89976137)

[5 Объектное моделирование на основе стандарта UML 25](#_Toc89976138)

[5.1 Диаграмма последовательности 25](#_Toc89976139)

[5.2 Диаграмма вариантов использования 26](#_Toc89976140)

[5.3 Диаграмма состояний 27](#_Toc89976141)

[5.4 Диаграмма классов 28](#_Toc89976142)

[5.5 Диаграмма компонентов 29](#_Toc89976143)

[5.6 Диаграмма развёртывания 30](#_Toc89976144)

[6 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 32](#_Toc89976145)

[6.1 Описание блок-схемы алгоритма 32](#_Toc89976146)

[6.2 Архитектура REST 32](#_Toc89976147)

[6.3 Паттерн MVC 34](#_Toc89976148)

[7 Руководство пользователя 35](#_Toc89976149)

[8 Тестирование разработанной системы 42](#_Toc89976150)

[Заключение 46](#_Toc89976151)

[Список использованных источников 47](#_Toc89976152)

[Приложение А (обязательное) Диаграмма последовательности 49](#_Toc89976153)

[Приложение Б (обязательное) Диаграмма последовательности 50](#_Toc89976154)

[Приложение В (обязательное) Диаграмма последовательности 51](#_Toc89976155)

[Приложение Г (обязательное) Диаграмма вариантов использования 52](#_Toc89976156)

[Приложение Д (обязательное) Диаграмма состояний 53](#_Toc89976157)

[Приложение Е (обязательное) Диаграмма классов 54](#_Toc89976158)

[Приложение Ж (обязательное) Диаграмма компонентов 55](#_Toc89976159)

[Приложение З (обязательное) Диаграмма развёртывания 56](#_Toc89976160)

[Приложение И (обязательное) Блок-схема алгоритма 57](#_Toc89976161)

[Приложение К (обязательное) Информационная модель системы 58](#_Toc89976162)

[Приложение Л (обязательное) Функциональная модель системы 59](#_Toc89976163)

# Введение

В современном мире в связи с занятостью людей всё большую актуальность приобретают приложения по планированию задач, так как люди предпочитают быстро отправить информацию через социальные сети вместо того, чтобы общаться вживую. Тема проекта является актуальной, так как в текущий век информационных технологий люди становятся всё более зависимыми от современных устройств и программ, в результате чего теряют возможность взаимодействия лицом к лицу и стараются окружить себя как можно большим количеством девайсов. Появление эпидемии коронавируса подчеркнуло преимущества цифровых средств коммуникации и обострило необходимость существования приложений, для которых живое общение не требуется.

На основе приведённых выше аргументов можно выделить цель создания проекта, заключающуюся в реализации программного средства для управления домашними задачами. Для поставленной цели можно разработать задачи её достижения:

* формирование требований к программе и составление диаграмм, отражающих основную функциональность системы;
* планирование распределения информации в таблицах базы данных;
* выбор языка программирования, технологий создания приложения, типа базы данных и системы управления.
* тестирование программы.

На основе поставленных задач можно выделить объект и предмет исследования.

Объект – домашние задачи.

Предмет – методы и алгоритмы создания веб-приложений для планирования домашних задач.

Программа была создана на объектно-орентированном языке программирования Java с помощью фреймворков Spring, Spring MVC, Spring Security, Spring Boot и СУБД PostgreSQL.

Работа является актуальной, так как удовлетворяет всем вышеперечисленным требованиям современного мира.

# Описание предметной области

Технологии лежат в основе устройств, которые люди используют ежедневно, и помогают им проводить досуг, работать и общаться. Ноутбуки, планшеты и смартфоны являются незаменимыми «инструментами», которые люди используют для обустройства своей жизни, обмена мыслями и эмоциями и для получения доступа к информации. Но в будущем нас ждет еще больше возможностей. В настоящее время происходит тенденция уменьшения размеров, увеличения производительности и внедрения интуитивно понятных пользовательских интерфейсов, что делает вычислительные устройства ещё более распространёнными в использовании.

Размер цифровых технологий имеет большое значение, так как чем меньше становятся процессоры, тем больше возможностей для их использования можно получить [2]. Уменьшение размеров процессоров и совершенствование свойства компактности ведёт к их внедрению практически в каждое вычислительное устройство.

В условиях развития информационного общества, в котором ключевым аспектом является создание глобального информационного пространства, обеспечивающего эффективное информационное взаимодействие людей, их доступ к мировым информационным ресурсам и удовлетворение их в информационных продуктах и услугах [3], простота использования должна быть основной отличительной особенностью всех технологий, делая взаимодействие пользователей с цифровыми устройствами простым и естественным. В условиях мобильного образа жизни люди должны иметь возможность управлять устройствами через касания и голосовые команды.

В процессе цифровой трансформации информация с бумажных носителей переносится на цифровые технологии [4]. Таким образом широкое распространение получают электронные блокноты, которые становятся заменой бумажных ежедневников. Они доступны на любых устройствах – от мобильных гаджетов до персональных компьютеров, всегда под рукой и одинаково хорошо подходят как для текстовых записей, так и для хранения голосовых заметок, фотографий и видео, составления списков дел, записи сиюминутных мыслей и свежих идей. С их помощью можно удобно упорядочивать и структурировать информацию, быстро находить нужные заметки, планировать задачи, создавать напоминания и объединять все записи в одном приложении, с выбором которого необходимо определиться.

Анализ планировщиков задач. Существует несколько критериев оценки программ для ведения заметок, которыми нужно руководствоваться в процессе выбора наиболее удобного цифрового пользовательского ежедневника.

Для многих пользователей важна поддержка облачных технологий, чтобы не держать в памяти гаджета или персонального компьютера программу, которая не умеет синхронизировать данные в рамках одной учётной записи между несколькими устройствами.

Также важным фактором является функциональность, то есть продукт должен в полной степени обеспечивать потребности рядовых пользователей и при этом не концентрироваться на бессмысленных настройках.

Стабильность работы и удобство интерфейса в сочетании с грамотным расположением элементов управления программой являются важными критериями для многих пользователей.

На рынке цифровых планировщиков задач существует большое количество приложений, наиболее востребованные из них будут приведены ниже [5].

OneNote. Это цифровая записная книжка [6], разработчиком которой является компания Microsoft, оно поддерживает следующие платформы: Web, Windows, macOS, Android, iOS, iPadOS.

OneNote позволяет создавать неограниченное количество записных книжек, которые, в свою очередь, можно разбивать на страницы и разделы – всё это помогает выстраивать многоуровневую структуру данных и грамотно организовывать информацию. Поддерживаются функции рукописного ввода и рисования, можно вставлять видео из Интернета и снимки экрана, записывать звуковые заметки, а также добавлять документы Word, таблицы Excel и презентации PowerPoint как в виде вложенных файлов, так и в формате виртуальных распечаток (изображений). Важные и актуальные записи можно помечать тегами для последующего быстрого доступа к ним, а для записных книжек с конфиденциальной информацией предусмотрена защита паролем. Также OneNote имеет развитые средства поиска и позволяет делиться заметками с сотрудниками, друзьями и близкими.

Из отличительных особенностей OneNote стоит выделить удобное ленточное меню, возможность писать в произвольном месте на листе и умение комбинировать рукописные фрагменты с печатными. Отдельного упоминания заслуживают встроенный OCR-движок, позволяющий извлекать тексты из изображений, и интеграция с сервисом перевода Microsoft Translator. Ну и наконец, интересная фишка программы – возможность решения математических уравнений (в том числе написанных от руки) с пошаговым объяснением вслух всех этапов решения задачи. Такая функция может послужить хорошим подспорьем для школьников старших классов и студентов. Microsoft активно продвигает OneNote в образовательной среде и предлагает бесплатные лицензии для школ и вузов, в программу добавлена возможность решения математических уравнений (в том числе написанных от руки) с пошаговым объяснением вслух всех этапов решения задачи, что может послужить в качестве помощника школьников и студентов.

Без оформления подписки на Microsoft 365 пользователю предлагается лишь базовый набор функций OneNote и 5 Гбайт облачного пространства для хранения данных. В бесплатном варианте не будет поддержки локальных записных книжек и распознавания рукописного ввода, а возможности решения математических уравнений будут сведены к минимуму.

Evernote. Это цифровой органайзер данных [7], разработчиком которого является компания Evernote Corporation, он поддерживает следующие платформы: Web, Windows, macOS, Android, iOS, iPadOS.

Данное приложение является одним из первых на рынке программного обеспечения для ведения заметок и упорядочивания информации. Evernote имеет широкую функциональность и ориентирован на корпоративный сектор.

По части предлагаемых возможностей Evernote поддерживает работу с заметками разного типа (текстовые, рисунки, фотографии, аудиофайлы, видео, PDF, копии веб-страниц, скриншоты и многое другое), позволяет прикреплять к записям документы Microsoft Office, PDF и другие файлы, умеет распознавать печатные документы, рисунки и рукописный текст. Заметки можно упорядочивать любым удобным способом, добавлять к ним метки (теги) и делиться записями с другими людьми.

Данная программа предоставляет возможности поиска текста в PDF и документах Office, сканирования и извлечения данных из визитных карточек, интеграции с Google Drive, Outlook, Slack, Microsoft Teams и другими сервисами, настройки разного уровня разрешений на доступ к содержимому и совместного редактирования заметок в режиме реального времени. Специально для корпоративных пользователей в программе предусмотрен набор шаблонов для быстрого создания записей подходящего формата и оформления.

Google Keep. Это приложение для создания заметок [8], разработчиком которого является компания Google, он поддерживает следующие платформы: Web, Android, iOS, Chrome OS.

Органайзер заметок, основные преимущества которого – высокая скорость работы, оптимальная функциональность и выверенный минимализм интерфейса без ненужных излишеств. В возможности приложения входит хранение текстов, аудиозаписей, фотографий, рисунков, списков задач, ссылок. Информация в программе записывается на отдельных заметках-стикерах, которые для упрощения поиска можно окрашивать в различные цвета и помечать тегами.

Keep поддерживает голосовой ввод текстовых данных, позволяет создавать напоминания, делиться информацией с другими пользователями, распознавать текст с фотографий печатных документов. Для удобства работы заметки можно сортировать по цвету, содержанию или наличию совместного доступа. Все функции реализованы максимально интуитивно и просто – в духе программных решений Google. Из замеченных особенностей Keep отметим существование в сервисе ограничения в 20 тысяч знаков (с пробелами) на размер заметок. Об этом моменте следует помнить пользователям, которые собираются оперировать большими объёмами данных или, например, писать при помощи Keep книгу.

Simplenote. Это система для создания и синхронизации заметок [9], разработчиком которой является компания Automattic, она поддерживает следующие платформы: Web, Windows, macOS, Linux, Android, iOS.

Приложение Simplenote позволяет работать только с текстами – поддержки иллюстраций, вложенных файлов и данных прочих форматов здесь нет, так что оно подойдёт пользователям, которые записывают строгий текст.

При внешней простоте интерфейса Simplenote достаточно функционален. Приложение синхронизирует книжки между устройствами, поддерживает взаимодействие с тегами и полнотекстовый поиск по базе данных, позволяет оперировать облегчённым языком разметки Markdown, допускает совместную работу над заметками и публикацию отдельных записей в Интернете для всеобщего обозрения. Помимо этого, Simplenote автоматически фиксирует все вносимые в заметки изменения и благодаря истории правок позволяет в любой момент откатывать их до одной из предыдущих версий.

Notion. Это рабочее пространство для создания заметок [10], разработчиком которого является компания Notion Labs, он поддерживает следующие платформы: Web, Windows, macOS, Android, iOS.

Notion является универсальной программой, которую в равной степени можно использовать и как приложение для хранения заметок, и в качестве платформы для создания баз знаний наподобие «Википедии», управления задачами и совместной работы с коллегами. Это аналог Evernote, в котором основной акцент сделан на возможностях подачи информации различными способами: форматируемыми в свободном стиле текстовыми блоками, таблицами, чек-листами или многоуровневыми списками, а также дизайнерскими шаблонами документов.

Notion довольно сложен для освоения без предварительного изучения документации и требует много памяти мобильных устройств.

Diigo. Это система для хранения данных [11], разработчиком которой является компания Diigo, она поддерживает следующие платформы: Web, Android, iOS.

Данный продукт изначально развивался как облачный сервис для хранения веб-страниц в закладках и перерос в более функциональное решение. На текущем этапе своего развития Diigo позволяет хранить не только URL и кэшированные копии веб-страниц, но и текстовые заметки, изображения и PDF-файлы. В составе программы представлен графический редактор для нанесения заметок на скриншоты и такой же редактор PDF-документов для выделения цветом важной информации и добавления примечаний. Предусмотрена возможность коллективной работы с хранимыми в Diigo данными и поддержка социальных групп (сообществ) для обмена закладками, заметками и изображениями.

Результаты анализа. Таким образом, можно сделать вывод о том, что на рынке программ по хранению заметок существует большое количество проектов. Каждый проект нацелен на решение своих определённых задач в зависимости от целевой аудитории, например, для людей в бизнес-сферах и для пользователей, которые предпочитают планировать свой день. Несмотря на то, что аналогов программ для записи заметок большое количество, конкретного проекта, нацеленного на семью, на рынке не существует.

Программное средство MyFam рассчитано на семьи, пары или людей, которые проживают в одном пространстве и является органайзером домашних задач. Оно позволяет создавать абстракцию дома с уникальным именем, добавлять в него комнаты, которые являются коллекциями домашних задач. Таким образом, заметки чётко структурированы в соответствии с комнатами, к которым они относятся, что делает проект наиболее быстрым и удобным для выполнения домашних дел.

# Постановка задачи и обзор методов её решения

## Постановка задачи

Задачей проекта является разработка программного средства для управления домашними задачами. Для достижения поставленной задачи можно выделить следующие подзадачи:

* разработать диаграммы для определения функциональности системы;
* создать иерархию классов, основываясь на спроектированной системе;
* спроектировать базу данных со всеми необходимыми таблицами;
* предусмотреть механизм защиты безопасности программного средства;
* обеспечить систему удобным графическим пользовательским интерфейсом;
* протестировать работу конечной системы.

Бизнес-логика приложения должна быть отделена от пользовательского интерфейса и реализована на серверной части. Работа в системе предусмотрена для двух ролей: пользователь и администратор.

Пользовательская часть должна предоставлять следующие возможности:

* регистрация и вход;
* создание и выбор дома;
* создание и выбор комнаты;
* добавление родственника в дом;
* добавление домашней задачи в комнату;
* управление домашними задачами, то есть их просмотр, удаление и отметка о завершении;
* добавление отзыва о работе приложения.

Администратор системы должен иметь следующие функции:

* переход на страницу пользователей;
* переход на страницу домов;
* вывод числа зарегистрированных пользователей;
* создание отчёта о работе приложения;
* управление аккаунтами зарегистрированных пользователей.

Необходимо обеспечить удобный интерфейс для взаимодействия пользователей и администратора с программным средством.

## Описание методов решения поставленной задачи

Исходя из определённых подзадач можно определить инструменты для создания программного средства.

Maven. Maven – инструмент для автоматизации сборки проектов на Java. Для сложных ПО с большим количеством подключаемых сторонних библиотек и ресурсов команда для сборки сложна, Maven разработан для упрощения процесса [12]. Одна из главных особенностей фреймворка – декларативное описание проекта. Это значит, что разработчику не нужно уделять внимание каждому аспекту сборки: все необходимые параметры настроены по умолчанию. Изменения нужно вносить в том объёме, в котором программист хочет отклониться от стандартных настроек.

Maven подгружает в свой локальный репозиторий сторонние библиотеки и выбирает необходимую версию пакета, обеспечивая таким образом гибкое управление зависимостями.

Ключевым понятием Maven является артефакт [13] – это любая библиотека, хранящаяся в репозитории, то есть зависимость или плагин.

Зависимости – это те библиотеки, которые непосредственно используются в проекте для компиляции кода или его тестирования.

Плагины используются Maven при сборке проекта, создании файлов проекта и других целей.

Архетип – это стандартная компоновка файлов и каталогов в проектах различного рода, исходя из особенностей их архитектуры.

POM (Project Object Model) является базовым модулем Maven [14]. Это специальный XML-файл, который всегда хранится в базовой директории проекта и называется pom.xml. Файл POM содержит информацию о проекте и различных деталях конфигурации, которые используются Maven для создания проекта. Этот файл также содержит задачи и плагины. Во время выполнения задач, Maven ищет файл pom.xml в базовой директории проекта. Он читает его и получает необходимую информацию, после чего выполняет задачи.

При необходимости систему сборки можно настроить, используя готовые плагины и архетипы, а также написать свои.

PostgreSQL. Для проектирования БД была выбрана СУБД PostgreSQL. Это объектно-реляционная СУБД, фундаментальной характеристикой которой является поддержка пользовательских объектов и их поведения [15], включая типы данных, функции, операции, домены и индексы, что делает данную СУБД достаточно гибкой и надёжной. Также PostgreSQL создаёт, хранит и извлекает сложные структуры данных, которые не поддерживаются стандартными РСУБД.

Spring. Spring framework – это платформа разработки интерпрайз-приложений на языке программирования Java с открытым исходным кодом, обеспечивающая продуманную модель программирования и конфигурации.

Spring предоставляет каркас, то есть задаёт правила построения приложения, его определённую архитектуру, в которую разработчику нужно встроить дополнительную функциональность: бизнес-логику приложения. В состав Spring входит много подпроектов, настроенных под определенную функциональность, напрмер, SpringMVC, Spring Security, SpringData и другие, из которых разработчик может выбрать наиболее подходящий ему, а остальные не использовать – в этом заключается модульный принцип построения приложения.

Spring Data. Spring Data – дополнительный механизм для взаимодействия с сущностями базы данных [16], организации их в репозитории, извлечения данных, и их изменения, для чего достаточно объявить интерфейс и метод в нем, без имплементации.

Spring Security. Spring Security это Java/JavaEE фреймворк, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Spring Security является стандартом при создании проектов с использованием Spring, это легко расширяемая под требования системы платформа, которая фокусируется на обеспечении как аутентификации, так и авторизации для приложений Java [17].

Spring Boot. Spring Boot – это проект, целью которого является упрощение создания приложений на основе Spring [18]. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов.

Таким образом, использование фреймворка Spring и его составляющих в сочетании с СУБД PostgreSQL предоставляют широкий набор инструментов для написания программного кода и являются оптимальными методиками для решения поставленной задачи и создания программного средства управления домашними задачами.

# Функциональное моделирование на основе стандарта IDEF0

Процесс бизнес-моделирования может быть реализован в рамках различных методик, отличающихся прежде всего своим подходом к тому, что представляет собой моделируемая организация. В соответствии с различными представлениями об организации методики принято делить на объектные и функциональные (структурные).

Объектные методики рассматривают моделируемую организацию как набор взаимодействующих объектов – производственных единиц. Объект определяется как осязаемая реальность – предмет или явление, имеющие четко определяемое поведение. Целью применения данной методики является выделение объектов, составляющих организацию, и распределение между ними ответственностей за выполняемые действия.

Функциональные методики, наиболее известной из которых является методика IDEF, рассматривают организацию как набор функций, преобразующий поступающий поток информации в выходной поток. Процесс преобразования информации потребляет определенные ресурсы. Основное отличие от объектной методики заключается в четком отделении функций (методов обработки данных) от самих данных.

С точки зрения бизнес-моделирования каждый из представленных подходов обладает своими преимуществами. Объектный подход позволяет построить более устойчивую к изменениям систему, лучше соответствует существующим структурам организации. Функциональное моделирование хорошо показывает себя в тех случаях, когда организационная структура находится в процессе изменения или вообще слабо оформлена. Подход от выполняемых функций интуитивно лучше понимается исполнителями при получении от них информации об их текущей работе.

Методологию IDEF0 можно считать следующим этапом развития хорошо известного графического языка описания функциональных систем SADT (Structured Analysis and Design Technique). Целью методики является построение функциональной схемы исследуемой системы, описывающей все необходимые процессы с точностью, достаточной для однозначного моделирования деятельности системы.

Модель IDEF0 начинается с представления системы как единого целого – одного функционального блока с интерфейсными дугами, простирающимися за пределы рассматриваемой области. Такая диаграмма с одним функциональным блоком называется контекстной диаграммой. На рисунке 3.1 представлена контекстная диаграмма объекта моделирования.

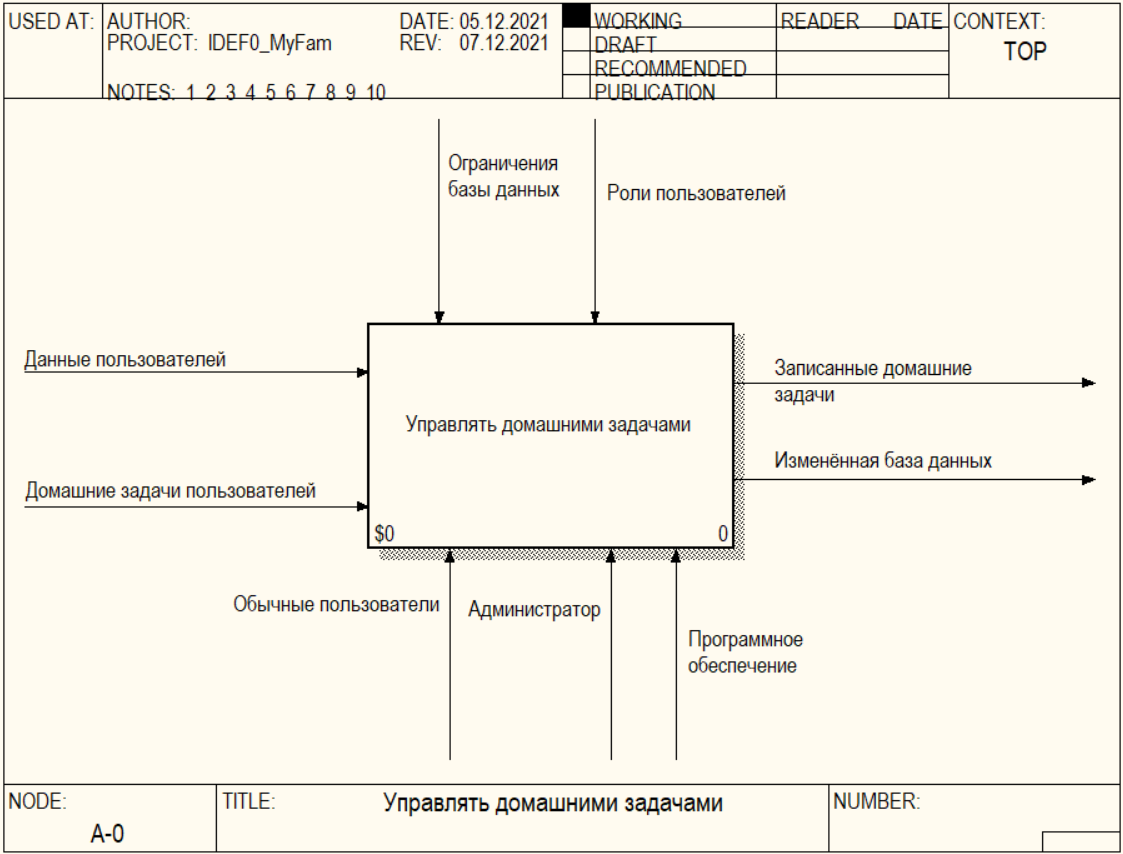


Рисунок 3.1 – Контекстная диаграмма управления домашними задачами

Контекстная диаграмма бизнес-процесса «Управлять домашними задачами» отображает систему в целом и ее взаимодействие с основными внешними потоками информации. В качестве входных данных используются домашние задачи и данные пользователей, поэтому на выходе системы представлены домашние задачи пользователей и изменённая база данных программного средства. В качестве управления в системе используются роли пользователей для защиты различных *URL* и ограничения базы данных, которые регулируют заполнение таблиц пользователями. Обычные пользователи, администратор и программное обеспечение являются механизмом системы.

Проведём декомпозицию контекстной диаграммы, описав последовательность управления домашними задачами, как показано на рисунке 3.2.

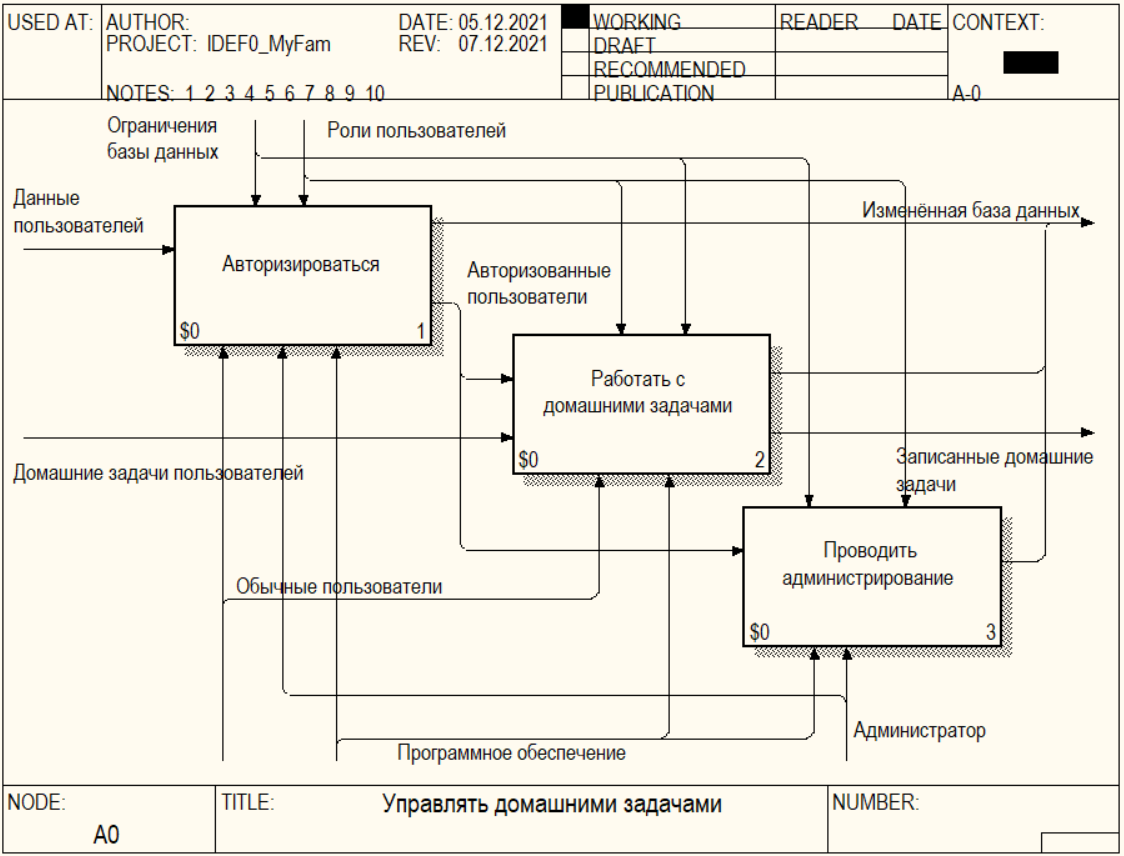


Рисунок 3.2 – Диаграмма декомпозиции управления домашними задачами

Диаграмма декомпозиции состоит из трёх блоков.

Первый блок отражает процесс определения уровня доступа в систему и определение роли пользователя. Далее проводится процедура доступа в систему, проверяются уникальное имя и пароль для выдачи роли и открытия доступа к следующим разделам сайта, которые представлены в виде блоков. Второй блок представляет процесс работы с домашними задачами, который может быть осуществлён только зарегистрированными в системе пользователями. Если роль пользователя – администратор, он может получить доступ к разделу администрирования на сайте. Как и на контекстной диаграмме, кроме входных и выходных ресурсов каждый блок имеет такие составляющие, как механизмы и управление.

Закончив декомпозицию контекстной диаграммы, можно перейти к декомпозиции диаграммы следующего уровня. Декомпозируем работу блока «Авторизоваться», как показано на рисунке 3.3

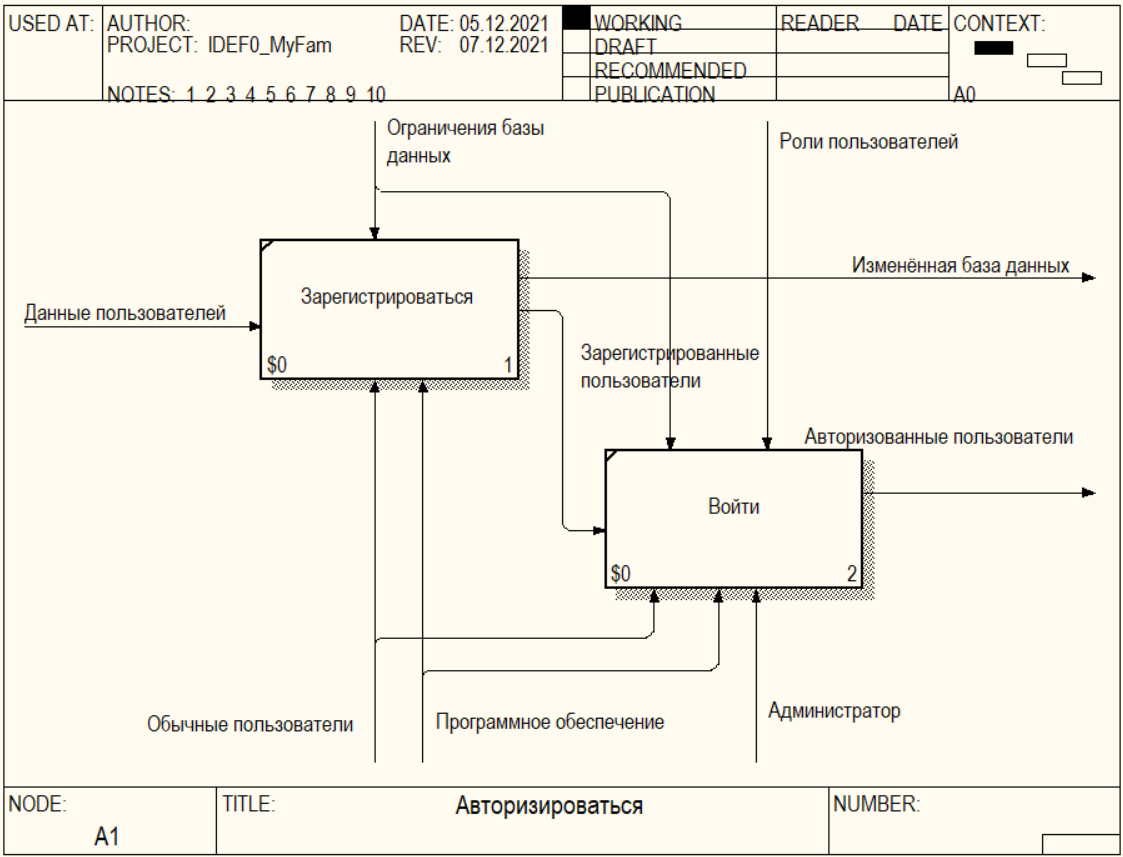


Рисунок 3.3 – Диаграмма декомпозиции блока авторизации

Блок авторизации раскладывается на две составляющие части, отражающие регистрацию и вход пользователя. Для регистрации любому посетителю сайта необходимо ввести свои уникальные данные и получить роль пользователя системы. После прохождения процедуры регистрации данные о новом пользователе записываются в базу данных, и ему предоставляется возможность входа в систему.

При осуществлении входа необходимо выделять три основных понятия: идентификация, аутентификация и авторизация.

Идентификация – процедура, в результате выполнения которой для субъекта идентификации выявляется его идентификатор, однозначно определяющий этого субъекта в информационной системе.

Аутентификация – процедура проверки подлинности, например проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.

Авторизация – предоставление определенному лицу или группе лиц прав на выполнение определенных действий.

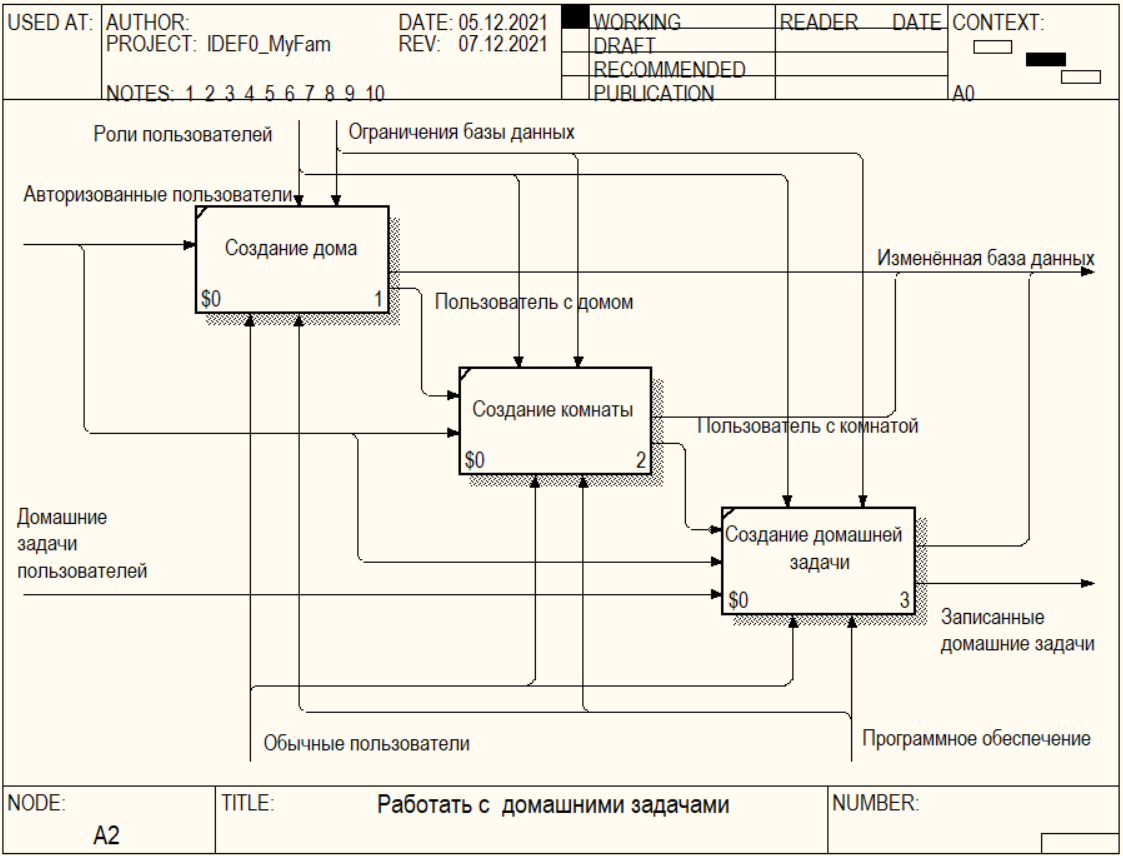


Рисунок 3.4 – Диаграмма декомпозиции блока работы с домашними задачами

На рисунке 3.4 представлена декомпозиция блока под названием «Работа с домашними задачами». В нём представлены три блока, показывающих процессы создания дома, создания комнаты и создание домашней задачи. Эти действия могут проводить только зарегистрированные на веб-сайте пользователи. После совершения каждой работы осуществляется процесс изменения базы данных.

На рисунке 3.5 представлена декомпозиция блока с названием «Проводить администрирование».

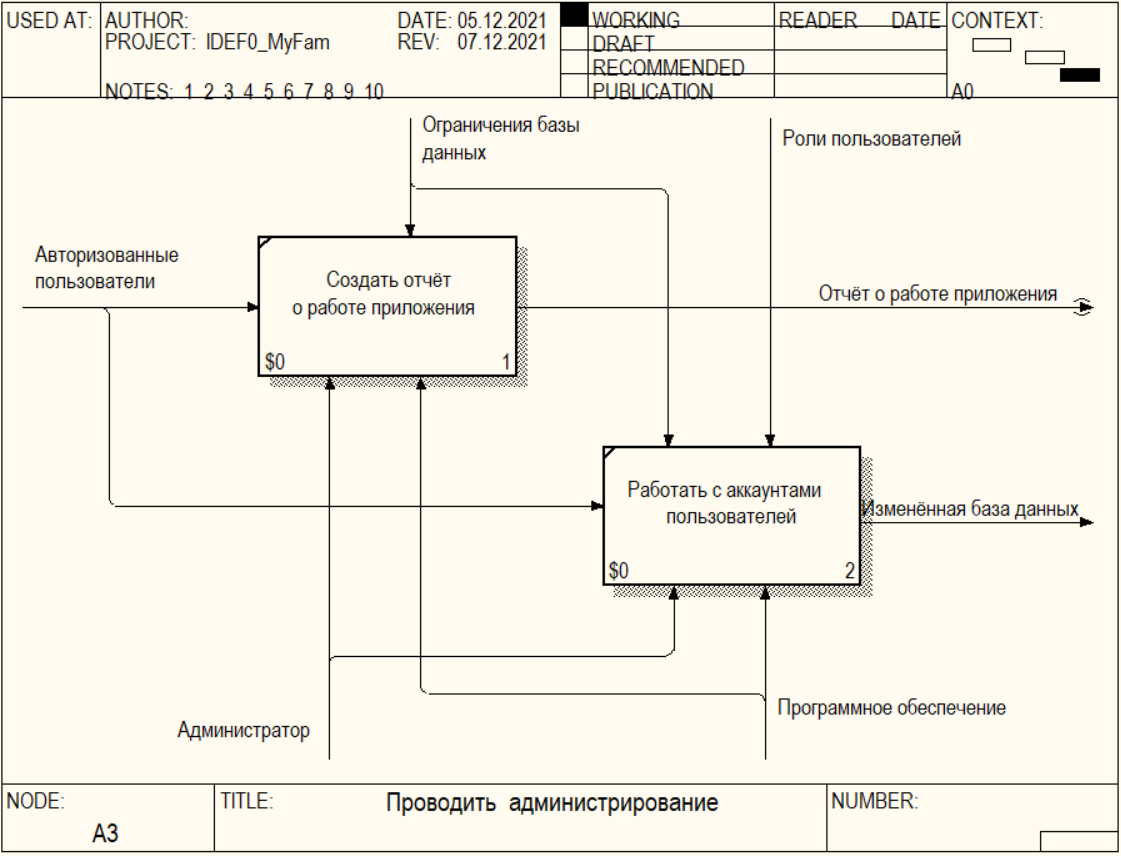


Рисунок 3.5 – Диаграмма декомпозиции блока администрирования

На этой диаграмме представлены два блока, которые отражают обязанности администратора сайта, заключающиеся в создании отчёта о работе приложения на основе отзывов пользователей и работе с аккаунтами обычных пользователей системы. Результатом первой работы является стрелка с названием «Отчёт о работе приложения», она туннелированная, так как не отображена на родительской диаграмме. Результатом второй работы является изменение базы данных.

Согласно описанию системы, её основной функцией является управление домашними задачами пользователей. Таким образом, система представлена в виде набора взаимосвязанных и взаимодействующих блоков, описанных в текущем разделе.

# Информационная модель системы и её описание

Методология *IDEF1X* используется для создания информационной модели в виде набора *ERD*-диаграмм, которые представляют собой структуру информации, необходимой для поддержки функций производственной системы или среды. Диаграммы «сущность-связь» предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними, детализация хранилищ данных проектируемой системы осуществляется с помощью *ERD* [19].

Проект использует пять связанных между собой таблиц, описание которых представлено ниже и одну таблицу для отзывов посльзователей.

Таблицы, отражённые на рисунке 4.1, созданы в реляционной базе данных с использованием системы управления базами данных *PostgreSQL*. Основные четыре таблицы созданы как сущности классов *DefaultUser*, *Home*, *Room* и *Note*, а пятая таблица является результатом связи.

Diagram

Description automatically generated

Рисунок 4.1 – Информационная модель системы

Сущность *DefaultUser* представляет пользователя приложения, имеет первичный ключ *id*, который генерируется автоматически с использованием технологии инкрементирования. Сущность переносится в таблицу *defaultuser* и на основе полей класса создаёт шесть дополнительных атрибутов, хранящих имя, уникальное имя, пароль, роль пользователя и информацию о его аккаунте.

Сущность *Home* является домом пользователя в приложении и имеет первичный ключ *home\_id*, который генерируется автоматически с использованием указанной выше технологии. Сущность переносится в таблицу home и на основе полей класса создаёт один дополнительный атрибут, хранящий уникальное название дома.

Так как у одного пользователя может существовать один дом и более, а у одного дома – один пользователь и более, между сущностями установлена связь «многие ко многим», в результате чего создана дополнительная таблица *built\_homes*, хранящая два ключа, по которым установлена связь между сущностями.

Сущность *Room* представляет комнату с первычным ключом *room\_id*, который также генерируется автоматически. Сущность переносится в таблицу room и на основе полей класса создаёт дополнительный атрибут – уникальное название комнаты.

Так как в одном доме может быть более одной комнаты (но одна комната является обязательной), а каждая из комнат имеет только один дом, между ними создаётся отношение «один ко многим», что показано с помощью стрелки.

Сущность *Note* является домашней задачей пользователя в приложении и имеет первычный ключ *note\_id*, который генерируется автоматически с использованием указанной выше технологии. Сущность переносится в таблицу note и на основе полей класса создаёт дополнительные атрибуты, хранящие уникальное название домашней задачи, её содержание и статус завершённости.

Одна комната хранит много домашних задач (или не хранит задачи, если они не созданы), а каждая из домашних задач может принадлежать только к одной комнате, таким образом связь «один ко многим» показана с помощью стрелки.

# Объектное моделирование на основе стандарта UML

## Диаграмма последовательности

Диаграммы последовательностей используются для детального описания логики сценариев использования. Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями [20]. Объекты обозначаются прямоугольниками, сообщения (вызовы методов) – линиями со стрелками, а возвращаемые результаты - пунктирными линиями со стрелками. Прямоугольники на вертикальных линиях под каждым из объектов показывают «время жизни» (фокус) объектов.

На диаграмме последовательности в приложении А отражена регистрация человека в системе. Посетитель сайта взаимодействует с интерфейсом приложения, который посылать сообщения контроллеру регистрации. Так как уже существующий пользователь не может зарегистрировать своё уникальное имя второй раз, а также существуют ограничения базы данных, контроллер посылает сообщение классу *UserDetails* и базе данных для отслеживания правильности регистрации. В случае успешной регистрации пользователь может перейти на страницу входа и войти в систему.

На диаграмме последовательности в приложении Б отражён вход посетителя сайта в систему. Так как войти может только зарегистрированный пользователь, он вводит свои данные в форму интерфейса приложения, который затем посылает всю введённую информацию на контроллер, который, в свою очередь, отправляет данные на сверку базе данных. Если пользователь найден, происходит процесс выдачи ему роли для входа на сайт, после чего открывается страница домов. Если же пользователь не найден, форма обнуляется для повторного ввода данных.

На диаграмме последовательности в приложении В отражены действия для создания дома в системе. Пользователь взаимодействует с интерфейсом приложения, где размещена форма для ввода уникального названия дома. Интерфейс приложения передаёт данные контроллеру, после чего последний обращается к базе данных для контроля ввода. На ввод установлена защита от попытки задания пустого имени для дома, а база данных контролирует уникальность названия. Если название дома не удовлетворяет вводу, человек не имеет ограничений и может повторять операцию любое количество раз. Если данные введены успешно, страница обновляется, и в списке домов появляется новый объект, после чего зарегистрированный пользователь может выбрать название дома из списка и перейти на следующую страницу.

## Диаграмма вариантов использования

Диаграмма вариантов использования (англ. *use-case diagram*) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей [21].

На диаграмме вариантов использования в приложении Г отражены два актёра: пользователь с ролью *PERSON* и администратор с ролью *ADMIN*. После ввода данных входа в систему каждый пользователь получает доступ к определённому *URL* в зависимости от роли, таким образом пользователь имеет перечисленные ниже возможности.

Регистрация в системе, которая включает создание уникального id, сверку данных с ограничениями валидации и поиск пользователя в базе даных. В случае, если данные не удовлетворяют критериям, пользователь не будет зарегистрирован.

Вход в систему, для чего автоматически происходит поиск пользователя в базе данных и отклонение в том случае, если пользователя не существует.

Зарегистрированный пользователь может создать дом, в результате чего база данных проверит дом на уникальность и ему будет присвоен *homeId*. После создания дома пользователь может выбрать его из списка и перейти на нужную страницу.

Обитатель дома добавляет других пользователь в дом, для этого происходит их поиск в базе данных.

В дом можно добавить комнату, её название будет доступно в виде ссылки, после нажатия на которую пользователь перейдёт к управлению заметками.

Зарегистрированный пользователь может добавить отзыв о работе сайта, который сохранится в базу данных и будет доступен администратору.

Администратор системы входит в служебную часть системы на основе получения разрешения после сверки с базой данных, в его обязанности входит управление аккаунтами пользователей и их домами, создание отчёта пользовательских отзывов.

В языке *UML* имеется несколько стандартных видов отношений между актёрами и вариантами использования: ассоциация, включение, расширение и обобщение. На диаграмме отражены следующие стрелки отношений: ассоциация для выражения взаимодействия актёра с отдельным вариантом использования; включение, для задания поведения, которое всегда вкючается в функциональность базового варианта использования; расширение для выражения функциональности, которая не обязательно задействуется базовым вариантом использования.

## Диаграмма состояний

Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы [22]. Данная диаграмма полезна при моделировании жизненного цикла объекта. От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

Диаграмма состояний имеет свою терминологию.

Начальное псевдосостояние – начальное состояние системы.

Конечное псевдосостояние – состояние, которое обозначает границы систем или подсистем.

Переход – перемещение из одного состояния в другое.

Состояние – выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым актёрами результатам.

В ситуации, когда несколько состояний имеют общие переходы и внутренние активности, их можно превратить в подсостояния, а общее поведение перенести в суперсостояние.

В приложении Д показана диаграмма состояний, которая отражает процесс создания нового дома. На этапе входа в суперсостояние должно быть выполнено действие по входу в систему, а на момент выхода из данного состояния должен быть создан дом.

Маркер истории указывает на то, что после закрытия и открытия страницы заново пользователь начнёт с отмеченного состояния.

Для осуществления перехода из состояния ввода названия дома в состояние подтверждения нового названия, необходимо, чтобы уникальное имя дома удовлетворяло критериям. В обратном случае произойдёт переход в состояние отклонения названия, которое не ведёт к конечному псевдосостоянию.

## Диаграмма классов

Диаграмма классов определяет типы классов системы и различного рода статические связи, которые существуют между ними. На диаграммах классов изображаются также атрибуты классов, операции классов и ограничения, которые накладываются на связи между классами [23].

Атрибуты описывают свойства объектов класса, за именем атрибута может следовать его тип и значение по умолчанию. Операция есть функция или преобразование. Операция может иметь параметры и возвращать значения.

Между классами существуют разные отношения. Ассоциация – представляет собой отношения между экземплярами классов. Каждый конец ассоциации обладает кратностью, которая показывает, сколько объектов, расположенных с соответствующего конца ассоциации, может участвовать в данном отношении. Агрегация – это ассоциация типа «целое-часть». Агрегация в UML представляется в виде прямой с ромбом на конце.

Композиция – это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Наследование – это отношение типа «общее-частное». Позволяет определить такое отношение между классами, когда один класс обладает поведением и структурой ряда других классов.

На диаграмме классов в приложении Е изображены связанные между собой классы, которые переходят в таблицы базы данных, а также их атрибуты и операции.

Так как класс *DefaultUser* содержит в себе список домов и класс *Home* содержит список людей, они зависимы друг от друга. По условию программы пользователь не может существовать без дома, поэтому классы имеют отношение композиции. Выбор кратности связи обусловлен отношениями «многие ко многим», а также тем, что в одном доме может быть один пользователь и более и один пользователе может иметь более одного дома.

Классы *Home* и *Room* имеют отношение агрегации, так как дом является контейнером для комнат. При этом дом может содержать нуль комнат и более, а каждая комната может находиться только в одном доме – это выражено связью «многие к одному».

Классы *Note* и *Room* также имеют отношение агрегации, так как комната содержит домашние задачи. Связь классов «один ко многим» и возможность удаления всех задач из комнаты определяют заданную кратность связи.

## Диаграмма компонентов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка. Интерфейсы на компонентных схемах показывают, как компоненты соединены друг с другом и взаимодействуют друг с другом [24].

Компонентные диаграммы используются в компонентно-ориентированных разработках для описания систем с сервис-ориентированной архитектурой. Они нацелены на то, чтобы показать структуру самого кода.

Диаграмма компонентов может использоваться для фокусировки на отношениях между компонентами, скрывая при этом детализацию спецификации, что помогает в информировании и разъяснении функций создаваемой системы заинтересованным сторонам.

Компоненты связываются через зависимости, когда соединяется требуемый интерфейс одного компонента с имеющимся интерфейсом другого компонента. Таким образом иллюстрируются отношения клиент-источник между двумя компонентами. Зависимость показывает, что один компонент предоставляет сервис, необходимый другому компоненту. Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу. Порт используется, когда компонент делегирует интерфейсы внутреннему классу. Стрелка зависимости также может использоваться для отражения связи между двумя компонентами.

Диаграмма компонентов на приложении Ж отражает разработанные в программном коде компоненты. Первым компонентом является интерфейс приложения, обозначение связи говорит о том, что для функционирования он требует интерфейс, который предоставляет компонент контроллер. Далее контроллер требует четыре интерфейса, с которыми он взаимодействует через серверные классы, предоставляющие основные методы для работы с объектами классов. Представленные классы обращаются к интерфейсам, являющимися репозиториями и взаимодействующим с базой данных.

## Диаграмма развёртывания

Диаграммы развертывания используются для визуализации аппаратных процессоров, узлов и устройств системы, каналов связи между ними и размещения программных файлов на аппаратном обеспечении.

Диаграмма развертывания – это тип UML-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их [25].

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами UML-диаграмм, которые в основном описывают логические компоненты системы.

Основные элементы диаграммы – это узлы и артефакты. Узел, представленный в виде куба, представляет собой физическую сущность, которая выполняет одну или несколько компонентов, подсистем или исполняемых файлов. Узел может быть аппаратным или программным элементом. Артефакты – это конкретные элементы, которые вызваны процессом разработки. Примерами артефактов являются библиотеки, архивы, конфигурационные файлы, исполняемые файлы и так далее. Коммуникационная ассоциация представлена сплошной линией между двумя узлами и показывает путь связи между узлами. Устройство – это узел, который используется для представления физического вычислительного ресурса в системе. Примером устройства является сервер приложений. Спецификации развертывания – это файл конфигурации, например текстовый файл или XML-документ. В нем описывается, как артефакт развертывается на узле.

В приложении З диаграмма развёртывания показана на диаграмме З и показывает связь между программным обеспечением и аппаратным обеспечением, то есть оборудованием. На её выделенных узлах показаны укрупнённые компоненты системы. На клиентской части такими элементами являются персональный компьютер пользователя и веб-браузер. Серверная часть представлена веб-сервером, внутри которого изображена база данных и контейнер классов программы. Серверная часть связывается с клиентской с помощью протокола *http*.

# Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы

## Описание блок-схемы алгоритма

Алгоритм – это система точных и понятных предписаний о содержании и последовательности выполнения конечного числа действий, необходимых для решения любой задачи данного типа [26]. Для реализации предметной области данного проекта разработаны правила и принципы поведения объектов и операций, что является бизнес-логикой приложения.

Полный алгоритм приложения начинается с блока данных и выводит страницу входа на экран. Следующим идёт блок процесса, который обрабатывает полученные данные и переходит на блок решения с условием для выдачи следующей страницы. Если пользователя не существует, ему предлагается страница регистрации, которая проверяет валидность введённых данных. В случае, если пользователь существует в системе, ему предлагается страница домов, на которой он может вступить в процесс добавления нового дома или перейти на страницу любого дома из уже созданных. В доме также есть возможность добавления комнаты либо выбора комнаты из заранее созданных вариантов. Для каждой комнаты предусмотрена возможность добавления, удаления и изменения статуса заметок, что является целью написания программного средства.

## Архитектура REST

*REST* (*Representational state transfer*) – это стиль архитектуры программного обеспечения для распределенных систем, таких как *World Wide Web*, который, как правило, используется для построения веб-служб [27]. Термин *REST* был введен в 2000 году Роем Филдингом, одним из авторов *HTTP*-протокола. Системы, поддерживающие *REST*, называются *RESTful*-системами.

Протокол *HTTP*. Когда пользователь вводит в браузере *URL*-адрес, на сервер отправляется запрос на веб-сайт, идентифицированный данным *URL*-адресом. Затем этот сервер формирует и выдает ответ. Важным является формат этих запросов и ответов, который определяется протоколом *HTTP* – *Hyper Text Transfer Protocol*. Таким образом, когда пользователь набирает *URL* в браузере, он отправляет запрос *GET* на указанный сервер. Затем сервер отвечает *HTTP*-ответом, который содержит данные в формате *HTML*, и браузер получает этот *HTML*-код и отображает его на экране.

Прокотокол *HTTP* обеспечивает базовый уровень для создания веб-сервисов, ниже будут перечислены несколько его ключевых абстракций.

Ресурс – это ключевая абстракция, на которой концентрируется протокол *HTTP*. Это вся информация, которая будет показана через приложение. При разработке *RESTful* сервисов необходимо сосредоточить свое внимание на ресурсах приложения. Способ, которым идентифицируется ресурс для представления, состоит в том, чтобы назначить ему *URI* – универсальный идентификатор ресурса. То есть каждая единица информации однозначно определяется глобальным идентификатором, таким как *URL*, то есть он является первичным ключом для единицы данных и имеет строго заданный формат.

Действие над данными задается с помощью методов: *GET* (получить), *PUT* (добавить, заменить), *POST* (добавить, изменить, удалить), *DELETE* (удалить). Таким образом, каждому действию *CRUD* (*Create*, *Read*, *Update*, *Delete*), характеризующему работу с базами данныхсоответствуют определённые методы.

Данный паттерн является интерфейсом управления информацией и основывается на передаче информации в её первоначальном виде, то есть не заворачивая в *XML* и другие форматы.

*REST* фокусируется на ресурсах и на том, насколько эффективно разработчик выполняет операции с ними, используя *HTTP*.

*HTTP* является основным строительным блоком *REST* сервисов. Это протокол, который используется для определения структуры запросов и ответов браузера, который имеет дело главным образом с ресурсами, доступными на веб-серверах. Ресурсы идентифицируются с помощью *URI*, а операции над этими ресурсами выполняются с использованием глаголов, определенных протоколом *HTTP*.

В текущем проекте все контроллеры помечены аннотаццией @*RestController*, а к каждому методу контроллера стоит одна из четырёх аннотаций в зависимости от того, какую функцию они выполняют: @*GetMapping* для вывода данных на экран, @*PostMapping* для отправки данных в базу данных, @*DeleteMapping* для удаления данных и @*PutMapping* для совершения операций изменения.

## Паттерн MVC

MVC – это шаблон программирования, который позволяет разделить логику приложения на три части. Модель получает данные от контроллера, выполняет необходимые операции и передаёт их в вид. Вид или представление получают данные от модели и выводит их для пользователя. Контроллер обрабатывает действия пользователя, проверяет полученные данные и передаёт их модели [28].

Этот паттерн разработки нужен для того, чтобы разделить логические части приложения и создавать их отдельно друг от друга, то есть писать независимые блоки кода, которые можно менять, не затрагивая другие.

Например, чтобы можно было переписать способ обработки данных, не меняя при этом способ их отображения. Это позволяет эффективно работать нескольким программистам, каждый занимается своим компонентом. При этом разработчику не нужно вникать в чужой код и его действия никак не повлияют на другие фрагменты приложения.

Если же говорить о приложениях, то компоненты будут следующие:

* вид – интерфейс;
* контроллер – обработчик событий, инициируемых пользователем (нажатие на кнопку, переход по ссылке, отправка формы);
* модель – метод, который запускается обработчиком и выполняет все основные операции (получение записей из базы данных, проведение вычислений).

Реализация паттерна MVC может отличаться в зависимости от задачи. Например, в веб-разработке модель и вид взаимодействуют друг с другом через контроллер, а в приложениях модель может сама уведомлять вид, что нужно что-то изменить. MVC – подход к проектированию приложения, который предполагает выделение кода в блоки модель, представление и контроллер. Контроллер обрабатывает входящие запросы. Модель достаёт из базы данных информацию, нужную для выполнения конкретных запросов. Представление определяет результат запроса, который получает пользователь.

В проекте созданы три пакета, которые необходимы для реализации паттерна MVC, это пакет models с классами, которые переносятся в сущности базы данных, controllers с пятью основными контроллерами и папка templates, предоставляющая страницы. Методы контроллеров инициализируют переменные класса ModelAndView, которые сохраняют нужную страницу (view) и все объекты, которые нужно на неё передать (model), далее созданная переменная возвращается в метод.

# Руководство пользователя

Программное средство MyFam рассчитано на семьи, пары или людей, которые проживают в одном пространстве и является органайзером домашних задач. Оно позволяет создавать абстракцию дома с уникальным именем, добавлять в него комнаты, которые являются коллекциями домашних задач. Таким образом, заметки чётко структурированы в соответствии с комнатами, к которым они относятся, что делает проект быстрым и удобным для выполнения домашних дел.

Для запуска программы необходимо открыть IDE Intellij IDEA, нажать на кнопку Run в верхней правой части экрана или использовать сочетание клавиш Shift + F10, дождаться загрузки проекта и перейти по адресу http://localhost:8080/.

После совершения указанных действий пользователю будет представлена начальная страница, которая представляет выбор из двух кнопок: регистрация и вход, что представлено на рисунке 7.1.

Text

Description automatically generated with low confidence

Рисунок 7.1 – Начальная страница

Если человек уже зарегистрирован в системе, ему не нужно создавать новый аккаунт, поэтому он должен нажать на кнопку входа и перейти на страницу.

В том случае, если пользователь зашёл в систему в первый раз или ещё создал аккаунт, ему нужно перейти на страницу регистрации, которая представлена на рисунке 7.2.

Graphical user interface, application

Description automatically generated

Рисунок 7.2 – Страница регистрации

Посетителю сайта представлены три поля для ввода, в которые ему нужно будет ввести имя, уникальное имя для видимости на сайте и пароль. В приложении предусмотрена обработка исключений, которая описана в разделе 8. Если регистрация прошла успешно, на экране появится надпись, продемонстрированная на рисунке 7.3.



Рисунок 7.3 – Сообщение об успешной регистрации

После просмотра сообщения об успешной регистрации необходимо нажать на кнопку входа и перейти на соответствующую страницу, которая представлена на рисунке 7.4.

Graphical user interface, text, application, chat or text message

Description automatically generated

Рисунок 7.4 – Страница входа

На странице входа представлены два поля ввода: уникальное имя пользователя и пароль. Если посетитель страницы имеет аккаунт в системе, при нажатии на кнопку входа он перейдёт на страницу домов, представленную на рисунке 7.5. В обратном случае на экране появится сообщение с ошибкой входа.

При переходе на страницу домов пользователю представляются все его дома в виде списка с невидимой прокруткой блока страницы, если количество домов больше шести. Только что зарегистрированному пользователю автоматически добавляется дом, вместо которого он может создать свой дом с уникальным названием, что отражено на рисунке 7.5. В случае успешного создания дома пользователь получит сообщение об успехе и увидит только что созданный дом на обновлённой странице. Выбор уникального имени застрахован от ошибок, что описано в главе 8.



Рисунок 7.5 – Страница домов

Имя каждого дома является ссылкой, при нажатии на которую будет открыта страница текущего дома, что можно увидеть на рисунке 7.6.

Graphical user interface, text, application

Description automatically generated

Рисунок 7.6 – Страница выбранного дома

Страница разделена на 3 части. Правая колонка отражает всех людей, которые живут в доме и управляют домашними задачами. В целях защиты личных данных пользователь не может присоединиться к дому самостоятельно, его добавляют пользователи, которые уже находятся в доме. Для добавления нового обитателя дома необходимо ввести его уникальное имя в правом нижнем углу страницы. Для ввода данных существует обработка исключений, которая представлена в разделе 8.

Левая колонка имеет название выбранного дома, в ней показаны все комнаты дома, а также есть возможность добавления новой комнаты, название которой должно быть уникальным в текущем доме. Для добавления комнаты нужно поставить курсор в поле в левом нижнем углу, ввести название и нажать на кнопку добавления. Список комнат обновится, и пользователю сразу же будет доступен переход по имени комнаты.

Название комнаты является ссылкой, после нажатия на которую человек увидит все добавленные для этой комнаты задачи, что можно увидеть на рисунке 7.7. Домашние задачи можно добавлять и удалять, а также отмечать, как выполненные.

Graphical user interface, text, application

Description automatically generated

Рисунок 7.7 – Страница комнаты с домашними задачами

Домашняя задача добавляется внизу страницы, записывается в базу данных и становится доступной для всех пользователей.

Веб-сайт имеет возможность добавления отзыва для зарегистрированных пользователей, для этого нужно нажать на страницу домов и перейти по ссылке внизу страницы. После этого пользователь сможет оставить отзыв о сайте, как показано на рисунке 7.8, который сохранится в базе данных и будет доступен администратору.

A picture containing icon

Description automatically generated

Рисунок 7.8 – Страница создания отзыва

У сайта есть только один администратор, который получает доступ к пользователям, домам и отзывам, что видно на рисунке 7.9.

Text, letter

Description automatically generated

Рисунок 7.9 – Страница отзывов у администратора

На рынке программ по хранению заметок существует большое количество проектов. Каждый проект нацелен на решение своих определённых задач в зависимости от целевой аудитории, например, для людей в бизнес-сферах и для пользователей, которые предпочитают планировать свой день. Несмотря на то, что аналогов программ для записи заметок большое количество, конкретного проекта, нацеленного на семью, на рынке не существует.

Программное средство MyFam обладает интуитивно понятным интерфейсом и предоставляет возможности быстрого взаимодействия между обитателями дома, таким образом, оно нацелено на людей всех возрастных категорий.

# Тестирование разработанной системы

Тестирование программного обеспечения (Software Testing) – это проверка соответствия между реальным и ожидаемым поведением программы, которая осуществляется на конечном наборе тестов, выбранном определенным образом [29].

Целью проведенного тестирования является общая проверка реального функционирования программного средства на соответствие предъявленным требованиям. Весь этап представляет собой выявление ошибок системы. Для этого создаются искусственные ситуации, которые могут возникнуть в период работы ресурса и анализируется «поведение» ресурса на предложенных условиях.

Для проверки данного программного средства были применены 5 типов тестирования:

* функциональное тестирование;
* тестирование графического интерфейса (GUI);
* тестирование удобства использования (Usability);
* кроссбраузерное тестирование.

Функциональное тестирование – важнейший этап тестирования, который нельзя заменить или пропустить. При функциональном тестировании проверяются:

* навигация;
* корректная динамичная работа выборок из списков;
* система поиска по отдельным критериям;
* возможность редактирования программного средства в качестве разработчика.

Цель функциональных тестов состоит в том, чтобы проверить каждую функцию программного приложения, предоставляя соответствующий ввод, проверяя выход в соответствии с функциональными требованиями.

Тестирование пользовательского интерфейса. Графический интерфейс пользователя – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Проверяется в целом общий вид приложения и в отдельности формы, расположенные на странице Тестирование удобства использования – тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя.

Удобство пользования программным средством определяется:

* наличием требуемой пользователю функциональности и ее работоспособностью;
* простотой использования продукта и скоростью обучения;
* количеством ошибок, которые совершают пользователи из-за непонимания.

В тестировании удобства и простоты пользования основное внимание обращено на визуальное оформление, навигацию, качественное отображение и доступность элементов пользовательского интерфейса, логичность и наличие обратной связи.

Кроссбраузерность это проверка на правильность (соответствие требованиям и стандартам) отображения и функционирования веб-приложения в разных браузерах и на разных операционных системах. В современном мире стандартизация принимает глобальные масштабы, а потому большинство популярных браузеров одинаково обрабатывают код. При этом необходимость в кроссбраузерном тестировании не исчезает, так как далеко не все проблемы решаются стандартизацией.

Что необходимо проверять при кроссбраузерном тестировании:

* функциональные возможности продукта, реализуемые на стороне клиента;
* правильность отображения элементов графики;
* шрифты и размеры текстовых символов;
* доступность и функциональность разнообразных форм, включая их интерактивность.

Тест – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

В таблице 8.1 представлены тесты для программного средства.

Таблица 8.1 – Тест кейсы программного средства

| № | Тест | Шаги | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Зарегистри-роваться | 1.Зайти на окно с формой регистрации.  2. Ввести данные в поля *«Username»*, *«Name»*, *«Password»*.  3. Отправить форму. | 1. Открывается форма для регистрации.  2. Введенные данные отображаются на форме.  3.1. При попытке ввода пустой строки, не соответствующих требованиям данных или существующих в системе данных, которые должны быть уникальными, система выдает ошибку.  3.2. При успешном завершении регистрации отображается окно подтверждения почты.  4. Пользователь вошел в систему. |
| 2 | Авторизация | 1.Зайти на окно авторизации.  2. Ввести данные в поля *«Login»* и *«Password»*.  3. Подтвердить отправку формы. | 1. Открывается форма для авторизации.  2. Введенные данные отображаются на форме.  3.1. Если такой учетной записи не существует, система выдает ошибку.  3.2. При корректных данных пользователь успешно авторизовался. |
| 3 | Просмотр домашних задач | 1. Перейти на вкладку комнаты. | 1. На экране отобразились домашние задачи комнаты |
| 4 | Создание домашней задачи | 1.Перейти на вкладку комнаты.  2. Ввести домашнюю задачу.  3. Подтвердить отправку формы. | 1. Отображаются все домашние задачи.  2. Введенные данные отображаются на форме.  3.1. При попытке ввода пустой строки система выдаёт ошибку.  3.2. При корректных данных домашняя задача успешно добавляется. |
| 5 | Просмотреть информацию о всех пользователях | 1. Перейти на вкладку пользователей дома. | 1. Отображается группа всех добавленных в дом жильцов. |
| 6 | Добавление пользователя в дом | 1. Перейти на вкладку пользователей дома.  2. Ввести уникальное имя нового пользователя.  3. Подтвердить отправку формы. | 1. Отображается группа всех добавленных в дом жильцов.  2. Введенные данные отображаются на форме.  3.1. При попытке ввода пустой строки или несуществующего пользователя система выдаёт ошибку.  3.2. При корректных данных домашняя пользователь успешно добавляется. |

Таким образом, тестирование системы является необходимой составляющей при создании программного средства, так как позволяет выявить соответствия и различия между реальным и ожидаемым поведением программы и установить правильность реализации требований.

На конечном этапе создания данного программного средства система была протестирована по двум основным критериям: удобство пользовательского интерфейса и защита от неправильного пользовательского ввода. Так как при попытке ввода пустых сторок, не существующих уникальных имён в случае добавления пользователя в дом, уже существующих данных в случае регистрации и не удовлетворяющих критериям валидации данных в случае регистрации, система извещает пользователя об ошибке и ограничивает его возможность перехода на защищённые страницы, программное средство прошло испытание тестирования. Удобство пользовательского интерфейса подкрепляется подписью для каждого ввода и появляющимися сообщениями, для ознакомления с полной работой программного приложения пользователь имеет возможность прочтения руководства, описанного в одном из разделов.

На основе успешного тестирования можно сделать вывод о том, что система удовлетворяет всем заданным требованиям, выполняет все свои задачи и является работоспособной.

# Заключение

На основе приведённых выше аргументов можно выделить цель создания проекта, заключающуюся в реализации программного средства для управления домашними задачами. Для поставленной цели можно разработать задачи её достижения:

На этапе изучения темы программного средства поставлены четыре основные задачи:

* формирование требований к программе и составление диаграмм, отражающих основную функциональность системы;
* планирование распределения информации в таблицах базы данных;
* выбор языка программирования, технологий создания приложения, типа базы данных и системы управления.
* тестирование программы.

В описании выполнения проекта подробное расписаны все шаги, осуществлённые для выполнения поставленных задач, которые включали разработатку диаграммы для определения функциональности системы; создание иерархии классов, основываясь на спроектированной системе; проектирование базы данных со всеми необходимыми таблицами; внедрение механизма защиты безопасности программного средства; обеспечение системы удобным графическим пользовательским интерфейсом; тестирование работы конечной системы.

Для реализации поставленных задач были использованы фреймвор для автоматизированной сборки проекта *Maven*, фреймворк для создания веб-приложений *Spring* с его подсоставляющими, в том числе *Spring* *Boot*, *Spring* *Security*, *Spring* *Data*.

Работа является актуальной, в связи с тем, что занятость людей двадцать первого века обязывает их делиться информацией через социальные сети вместо общения лицом к лицу и планировать свои дела с помощью электронных ежедневников. Приложение *MyFam* удовлетворяет всем потребностям, так как является электронным программным средством для управления домашними задачами, что включает их создание и удаление, а также изменение статуса на завершение.

# Список использованных источников

[1] Век информационных технологий [Электронный ресурс]. – Режим доступа: https://refdb.ru/look/2451574.html

[2] Технологии в современном мире [Электронный ресурс]. – Режим доступа: https://www.iksmedia.ru/articles/5012696-Kak-texnologii-menyayut-nashu-zhizn.html

[3] Информационное общество [Электронный ресурс]. – Режим доступа: https://dic.academic.ru/dic.nsf/fin\_enc/23453

[4] Процесс цифровизации [Электронный ресурс]. – Режим доступа: https://secretmag.ru/enciklopediya/chto-takoe-cifrovizaciya-obyasnyaem-prostymi-slovami.htm

[5] Приложения для записи задач [Электронный ресурс]. – Режим доступа: https://3dnews.ru/1014032/obzor-10-programm-dlya-zametok

[6] OneNote [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/ru-ru/microsoft-365/onenote/digital-note-taking-app?ms.url=onenotecom&rtc=1

[7] Органайзер данных EverNote [Электронный ресурс]. – Режим доступа: https://evernote.com/

[8] Google Keep [Электронный ресурс]. – Режим доступа: https://www.google.ru/keep/

[9] Приложение Simplenote [Электронный ресурс]. – Режим доступа: https://simplenote.com/

[10] Рабочее пространство Notion [Электронный ресурс]. – Режим доступа: https://www.notion.so/

[11] Сборщик ресурсов Diigo [Электронный ресурс]. – Режим доступа: https://www.diigo.com/

[12] Особенности Maven [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/maven-short-intro/

[13] Ключевые понятия Maven [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/77382/

[14] Структура Maven [Электронный ресурс]. – Режим доступа: https://proselyte.net/tutorials/maven/pom/

[15] СУБД PostgreSQL [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/282764/

[16] Spring Data [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/435114/

[17] Основы Spring Security [Электронный ресурс]. – Режим доступа: https://spring.io/projects/spring-security

[18] Основы Spring Boot [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/435144/>

[19] Использование методологии IDEF1X [Электронный ресурс]. – Режим доступа: https://studme.org/87187/ekonomika/metodologiya\_idef1x

[20] Построение диаграммы последовательности [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_sequence-diagram.html

[21] Построение диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_use-case-diagram.html

[22] Построение диаграммы состояний [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_statechart-diagram.html

[23] Построение диаграммы классов [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/gpg\_class-diagram.html

[24] Построение диаграммы компонентов [Электронный ресурс]. – Режим доступа: https://creately.com/blog/ru/uncategorized-ru

[25] Построение диаграммы развёртывания [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd\_deployment-diagram.html

[26] Назначение алгоритма [Электронный ресурс]. – Режим доступа: http://psk68.ru/files/metod/uchebnik\_Informatika/algor.html

[27] Архитектура REST [Электронный ресурс]. – Режим доступа: https://javarush.ru/groups/posts/2486-obzor-rest-chastjh-1-chto-takoe-rest

[28] Паттерн MVC [Электронный ресурс]. – Режим доступа: https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami

[29] Тестрирование приложения [Электронный ресурс]. – Режим доступа: https://pvs-studio.com/ru/blog/terms/0093/

# Приложение А (обязательное) Диаграмма последовательности

# Приложение Б (обязательное) Диаграмма последовательности

# Приложение В (обязательное) Диаграмма последовательности

# Приложение Г (обязательное) Диаграмма вариантов использования

# Приложение Д (обязательное) Диаграмма состояний

# Приложение Е (обязательное) Диаграмма классов

# Приложение Ж (обязательное) Диаграмма компонентов

# Приложение З (обязательное) Диаграмма развёртывания

# Приложение И (обязательное) Блок-схема алгоритма

# Приложение К (обязательное) Информационная модель системы

# Приложение Л (обязательное) Функциональная модель системы