Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  старший преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.М. Унучек |
|  | \_\_\_.\_\_\_\_.2021 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«АВТОМАТИЗИРОВАННАЯ система ПОДДЕРЖКИ ПРОЦЕССОВ управления требованиями В it-проектах»**

БГУИР КП 1-40 01 02-08 08 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 910101  МАЗУР Ксения Андреевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку 12.2021  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc89967060)

1 ОПИСАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ……………4

1.1 Общие понятия и требования…………………….………....……..….....4

1.2 Цель разработки системы управления…………….........……….............8

[2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ 9](#_Toc89967061)

[3 ФУНКЦИОНАЛЬНАЯ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEFO 12](#_Toc89967062)

[4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ 15](#_Toc89967063)

[5 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ 17](#_Toc89967064)

[5.1 Диаграмма вариантов использования 17](#_Toc89967065)

[5.2 Диаграмма состояний 18](#_Toc89967066)

[5.3 Диаграмма последовательностей 19](#_Toc89967067)

[5.4 Диаграмма классов 20](#_Toc89967068)

[5.5 Диаграмма развертывания и компонентов 21](#_Toc89967069)

[6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ 23](#_Toc89967070)

6.1Алгоритм удаления требования……………………………………......23

6.2 Алгоритм добавления требования……………………………………..25

[7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ 26](#_Toc89967071)

[8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ 28](#_Toc89967072)

[ЗАКЛЮЧЕНИЕ 34](#_Toc89967073)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc89967074)

[ПРИЛОЖЕНИЕ А 35](#_Toc89967075)

# **ВВЕДЕНИЕ**

Несмотря на постоянный и значительный прогресс, наблюдаемый в индустрии программного обеспечения (ПО), у многих компаний-разработчиков по-прежнему возникают трудности при выявлении, сборе, документировании и управлении требованиями к ПО. Из-за не полностью сформулированных требований, их изменения, а также недостаточного объема поступающей от пользователей информации, компаниям-разработчикам не всегда удается вовремя выполнить работу и предоставить клиентам требуемую функциональность. Управление требованиями является одной из важнейших стадий системного проектирования. Требования позволяют получить максимально наглядное и согласованное представление о продукте, сократить цикл разработки и время выхода на рынок.

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, обычно очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно четко сформулировать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на программную систему, называется *требованиями к системе,* а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений — *разработкой требований* (requirements engineering).

Согласно стандарту РМ ВОК требования — это:

• условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;

• структурированные (уточненные) и задокументированные потребности, пожелания и ожидания заказчика, пользователя и других заинтересованных сторон.

Согласно IEEE Standard требование — это:

• возможность, необходимая пользователю для решения проблем или достижения целей;

• возможность, которой должна обладать система, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам.

Таким образом, *программные требования* (Software Requirements) — свойства ПО, которые должны быть надлежащим образом представлены для решения конкретных практических задач. Опыт индустрии информационных технологий показывает, что вопросы, связанные с управлением требованиями, оказывают критически важное влияние на программные проекты и, в определенной степени, на сам факт возможности успешного их завершения. Только систематичная работа с требованиями позволяет корректным образом обеспечить моделирование задач реального мира и формулирование необходимых приемочных тестов для того, чтобы убедиться в соответствии создаваемых программных систем критериям, заданным реальными практическими потребностями. [1]

Для достижения поставленной цели требуется решить следующие задачи:

* проанализировать заданную предметную область;
* рассмотреть физическую и логическую модель представления данных;
* создать базу данных;
* реализовать серверную часть, имеющую доступ к базе данных и выполняющую ряд необходимых действий, позволяющих достигнуть цель курсового проекта;
* реализовать клиентскую часть с дружественным интерфейсом, обеспечивающий максимально удобное взаимодействие с программой или вычислительной системой.

1. **ОПИСАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ**

**1.1 Общие понятия и требования**

Требования являются основой любого продукта. Они определяют потребности заинтересованных лиц и разработчиков и функционал, которым система должна обладать, чтобы удовлетворить эти потребности. Заинтересованными лицами являются личности, на которых оказывает влияние разрабатываемая система, обычно под ними подразумеваются пользователи и заказчики системы. Требования к продукту принято разделять на три уровня: бизнес-требования, требования пользователей и функциональные требования.

1) *бизнес-требования.* Данный вид требований формируется заказчиком ИС и основывается, прежде всего, на целях создания заказываемого продукта. Бизнес-требования определяют, какие преимущества должен получить заказчик при получении готового продукта, а также какие проблемы или задачи будут решены в результате его применения. В результате формулирования бизнес-требований очерчиваются границы создаваемой ИС, а также создается общий образ проекта. Например, на уровне бизнес- требований могут быть сформулированы требования к поддержке бизнес- процессов. Если в качестве проекта выступает CRM-система, то в качестве бизнес-требований к ней будут выделены коммуникационные, отчетные и управленческие процессы;

2) *пользовательские требования* — это задачи, которые будет решать ИС для поддержки пользователей. Функциональные требования данного уровня представляются в виде сценариев (user journey), алгоритмов и таблиц «событие — отклик». Также формирование пользовательских требований может вестись на основе ключевых ролей, которые будут использоваться для работы ИС. Возможности каждой роли, будь то «Клиент», «Инвестор», «Партнер» и др., будут различаться в дальнейшем;

3) *функциональные требования* — это основные требования по функциональности ИС, которые далее детально описываются в виде технического задания и передаются на реализацию разработчикам.

Но, как показала практика, одних только функциональных требований к ИС недостаточно, поскольку созданная лишь на их основе ИС не будет удовлетворять всем требованиям бизнеса. В связи с этим К. Вигерс выделяет три вида нефункциональных требований к ИС:

1) *бизнес-правила*, включающие требования регуляторов (например, экологические нормативы или нормы безопасности), промышленные стандарты, корпоративные стандарты и другие ограничения, которые неизбежно налагаются внешней средой или политикой компании;

2) *атрибуты качества*, которые не относятся к функциональности системы, однако являются обязательным условием для эффективного применения создаваемой ИС в дальнейшем. В качестве требований вида «Атрибуты качества» может выступать возможность интеграции с другими ИС, интероперабельность, поддержка программных продуктов и т.п.;

3) *ограничения*, к которым, как правило, относятся вынужденные технические или ресурсные ограничения (уровни производительности, технические протоколы и пр.). [2]

Разработка требований к ПО — процесс выявления, формулирования, анализа, документирования и верификации требований, подлежащих выполнению в продукте (ПО). В его ходе системный аналитик формирует реестр требований, который ложится в документ или автоматизированную систему управления требованиями [3]. Разработку требований принято считать самой сложной частью проектирования ПО. Этот этап подразделяется на извлечение, анализ, документирование и проверку требований. Исследование требований циклично, на каждой итерации происходит детализация высокоуровневых требований и подтверждение правильности требований будущими пользователями системы [4]. Управление требованиями является непрерывным процессом на протяжении всего проекта разработки программного обеспечения. Цель управления требованиями состоит в гарантировании, что организация документирует, проверяет и удовлетворяет потребности и ожидания её клиентов и внутренних или внешних заинтересованных лиц. Управление требованиями включает поддержку требований, интеграцию требований и организацию работы с требованиями и сопутствующей информацией, поставляющейся вместе с требованиями [5].

Для структурированного хранения требований и их детализации используют соответствующие инструментальные средства – системы управления требованиями (СУТ). СУТ имеют важные ключевые особенности: возможность указания связей между требованиями; возможность построения выборок в различных представлениях; функции отслеживания изменений. Этот базовый набор функций дает возможность отслеживать и контролировать ошибки проектирования, сокращать время и повышать точность формирования оценки изменений.[6] В настоящее время широкое распространение получили такие СУТ, как IBM Rational RequisitePro, Telelogic DOORS и Borland Caliber RM. Также для управления требованиями в процессе разработки ПО можно использовать легковесные веб-системы управления проектами, например, Redmine, Trello и т.д. Далее приведены некоторые особенности и сравнительный анализ подобных систем. IBM Rational RequisitePro – средство управления требованиями к ПО при разработке программного обеспечения. Оно позволяет разработчикам определять и управлять требованиями, систематизировать и отслеживать изменения, которые могут возникнуть на протяжении всего жизненного цикла проекта, создавать с помощью RationalRequisitePro качественные сценарии использования, организовывать и повышать эффективность совместной работы. IBM Rational/Telelogic DOORS — это семейство решений для управления требованиями, которое позволяет оптимизировать обмен информацией о требованиях, контролировать большой объем взаимосвязанной информации, обеспечивает проверку выполнения требований и управление ими. DOORS успешно используется при создании сложных наукоемких изделий (авиа, судостроение, поезда, ракеты, автомобили т.п.). Borland Caliber RM – корпоративная система управления требованиями, которая разработана для повышения качества создаваемых продуктов, путем улучшения взаимодействия между участниками команды, упрощения анализа влияний требований и процесса передачи информации и возможности непрерывного сбора пожеланий заинтересованных в проекте лиц на всех этапах жизненного цикла проекта. Redmine – это одна из наиболее популярных и современных систем управления задачами. Это открытое серверное веб-приложение для управления проектами и задачами, написанное на Ruby. Redmine представляет собой приложение на основе веб-фреймворка RubyonRails и распространяется согласно GeneralPublicLicense. Система позволяет проводить мониторинг и контроль выполнения задач. В отличии от представленного выше ПО для управления требованиями Redmine является легковесной и бесплатной системой, которая не требует выделенного или виртуального хостинга.

Каждая система обладает определенными достоинствами и недостатками. Для использования в крупных и сложных проектах становится рациональным использование коммерческих СУТ. При грамотном использовании они позволяют легче обнаруживать ошибки на ранних этапах проектирования и оценивать влияние изменений на систему, возможность их реализации, сроки и стоимость. Однако в небольших или некоммерческих проектах бывает удобнее и выгоднее использовать бесплатные и легковесные системы, такие как Redmine.

* 1. **Цель разработки системы управления**

Цель управления требованиями состоит в том, чтобы гарантировать, что организация документирует, проверяет и удовлетворяет потребности и ожидания её клиентов и внутренних или внешних заинтересованных лиц. Управление требованиями начинается с выявления и анализа целей и ограничений клиента. Управление требованиями, далее, включает поддержку требований, интеграцию требований и организацию работы с требованиями и сопутствующей информацией, поставляющейся вместе с требованиями.

Установленная таким образом отслеживаемость требований используется для того, чтобы уведомлять заинтересованных участников об их выполнении, с точки зрения их соответствия, законченности, охвата и последовательности. Отслеживаемость также поддерживает [управление изменениями](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8) как часть управления требованиями, так как она способствует пониманию того, как изменения воздействуют на требования или связанные с ними элементы, и облегчает внесение этих изменений.

Главной целью любой системы автоматизации является сокращение издержек и увеличение производительности работы организации. Также использование такой системы должно улучшить контроль над исполнением требований к ПО, что в свою очередь увеличит эффективность управления компанией, эффективность её работы.

Таким образом, необходимо создать программный продукт, который будет отвечать следующим требованиям:

* сотрудник должен затрачивать минимальное количество временных ресурсов на различные различные действия с требованиями;
* интерфейс должен быть удобным и понятным для пользователя;
* должна быть создана база данных для быстрого и удобного поиска интересующей информации.

# **ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ**

Для решения поставленных задач в курсовом проекте используются ниже описанные программные и технические средства.

Java – полностью объектно-ориентированный язык программирования. Java используется для разработки программного обеспечения проектов различного назначения: начиная от приложений для мобильного телефона, заканчивая масштабными корпоративными приложениями.

Встроенная поддержка многопоточности снабжает программистов Java мощным инструментом для улучшения интерактивной работы графических приложений. Потоки иногда также называют легкими процессами или контекстами выполнения. Библиотека Java обеспечивает класс потока, который содержит обширную коллекцию методов запуска, выполнения, и остановки потока, а также проверки его состояния.

Самой выдающейся чертой языка Java является то, что его приложения компилируются в специальный байт-код и выполняются в специальной виртуальной машине в любой операционной системе и на любом оборудовании.

Другим важным преимуществом языка является довольно высокая безопасность. Поскольку все приложения выполняются и, соответственно, контролируются Java-машиной, то при любой попытке несанкционированного доступа или соединения программа просто прерывает свою работу. Некоторые программисты выделяют и более упрощённую саму форму написания программ Java по сравнению с её предшественниками, коими являются С, С+ и С++. Помимо этого к языку существует огромное количество бесплатных программных библиотек, написанных программистами всего мира.

Также Java обеспечивает автоматическое освобождение неиспользуемой памяти. Наряду с распределением памяти программистом JVM ведет учет всех используемых блоков памяти и указателей на них. Если блок памяти больше не используется, он может быть освобожден. Это обеспечивает процесс, который называется "сборкой мусора". Он периодически вызывается JVM, проверяет все используемые блоки памяти и освобождает те из них, на которые отсутствуют указатели. Сборка мусора очень удобна, но за ее использование приходится расплачиваться большим потреблением памяти и низкой производительностью.

Данный курсовой проект реализован с помощью клиент-серверной архитектуры.

Технология «клиент-сервер» дает большую безопасность для пользователя, процессы выполняются более стабильно, согласованно, с повышенной конфиденциальностью, данные быстро обрабатываются и надежно хранятся.

В архитектуру клиент-сервер входят следующие основные компоненты:

* сервер баз данных отвечает за хранение, доступ, защиту и резервное копирование данных;
* сервер приложений - это устройство, выполняющее определенные бизнес-правила;
* клиент предоставляет интерфейс пользователя;
* сеть и коммуникационное ПО – это всевозможное оборудование, каналы для передачи данных и ПО используемое для осуществления передачи запросов и ответов от клиента к серверу и обратно через сетевые протоколы.

Использование такой архитектуры помогает оптимизировать распределение вычислительных ресурсов и обеспечивает защиту данных.

Как правило, программа обработки данных (клиентская часть) располагается на одном персональном компьютере, а сама база данных - на другом. Такое разделение вполне закономерно: клиентская часть программы (та, что находится у пользователя), передает по сети запрос на обработку данных на другой компьютер, на котором сервер базы данных их читает, выполняет, и возвращает по сети ответ клиенту. Причем по сети передается лишь полезная информация.

JavaFX представляет инструментарий для создания кроссплатформенных графических приложений на платформе Java.

JavaFX позволяет создавать приложения с богатой насыщенной графикой благодаря использованию аппаратного ускорения графики и возможностей GPU.

С помощью JavaFX можно создавать программы для различных операционных систем: Windows, MacOS, Linux, Android, iOS и для самых различных устройств: десктопы, смартфоны, планшеты, встроенные устройства, ТВ. Приложение на JavaFX будет работать везде, где установлена исполняемая среда Java (JRE).

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое. [7]

MySQL – это система управления реляционными базами данных.

В реляционной базе данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

MySQL поддерживает параллельный доступ к базе данных. Приложения могут обращаться к базе данных одновременно, что повышает общую производительность системы. Кроме того, отдельные операции могут «распараллеливаться» для еще большего улучшения производительности.

MySQL помогает восстанавливать информацию в случае непредвиденного сбоя, незаметно для пользователей создавая резервные копии данных. Все изменения, вносимые в базу данных, регистрируются, поэтому многие операции можно отменять и выполнять повторно.

# 

# **ФУНКЦИОНАЛЬНАЯ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEFO**

Целью данного курсового проекта является создание программного продукта, который автоматизирует процесс управления требованиями. IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов.

Ключевым процессом в данной теме является управление требованиями проекта. На контекстной диаграмме верхнего уровня (рисунок 3.1) представлена функциональная модель «Управление требованиями проекта», а также определены потоки входных и выходных данных, механизмы ограничения и управления данными.

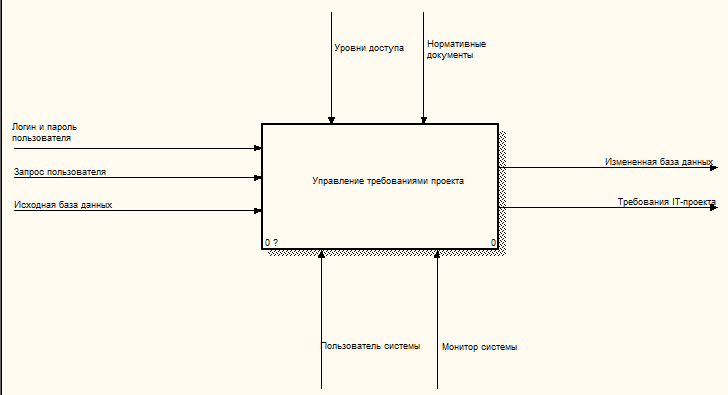


Рисунок 3.1– Контекстная диаграмма верхнего уровня

В основном процессе можно выделить следующие подпроцессы:

* определение уровня доступа в систему;
* обращение к системе;
* изменение базы дынных;
* обработка запроса пользователя.

Декомпозиция контекстной диаграммы верхнего уровня показана на рисунке 3.2.

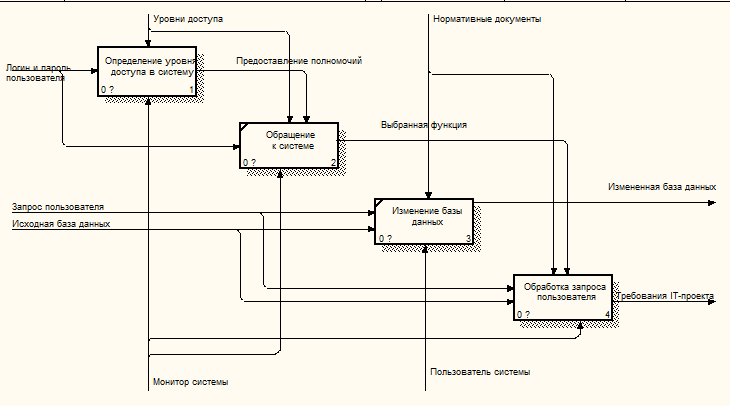


Рисунок 3.2– Декомпозиция контекстной диаграммы верхнего уровня

Декомпозиция блока «Обработка запроса пользователя» представлена на рисунке 3.3.

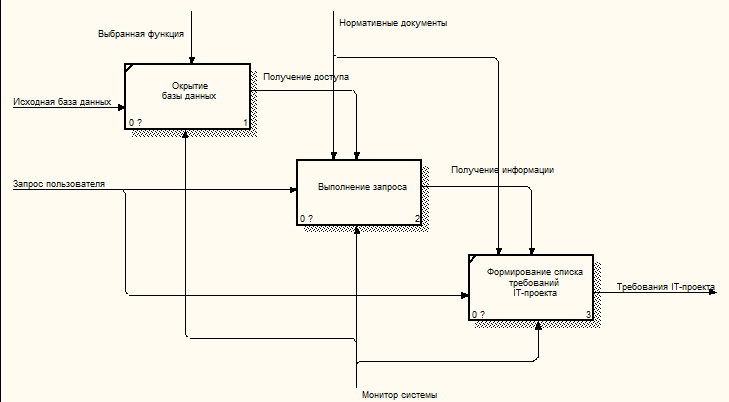


Рисунок 3.3– Декомпозиция блока «Обработка запроса пользователя»

Декомпозиция блока «Определение уровня доступа в систему» представлена на рисунке 3.4.

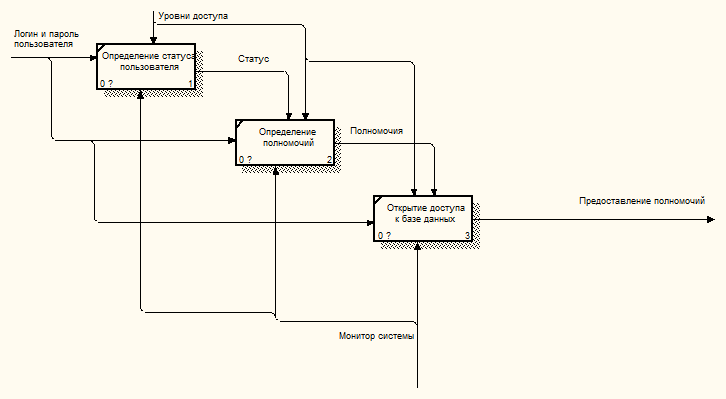


Рисунок 3.4– Декомпозиция блока «Определение уровня доступа в систему»

Декомпозиция блока «Выполнение запроса» представлена на рисунке 3.5.

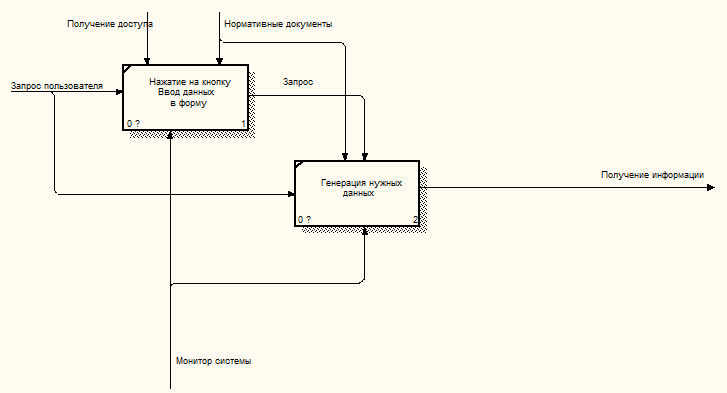


Рисунок 3.5 – Декомпозиция блока «Выполнение запроса»

# **ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ**

Информационная модель в данном курсовом проекте была построена с помощью средства AllFusion ERwin Data Modeler.

AllFusion ERwin Data Modeler – [CASE](https://ru.wikipedia.org/wiki/CASE)-средство для [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [документирования](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F) [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), которое позволяет создавать, документировать и сопровождать базы данных, хранилища и витрины данных. [Модели данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) помогают визуализировать структуру данных, обеспечивая эффективный процесс организации, управления и администрирования таких аспектов деятельности предприятия, как уровень сложности данных, технологий баз данных и среды развертывания.

AllFusion ERwin Data Modeler позволяет наглядно отображать сложные структуры данных. Удобная в использовании графическая среда AllFusion ERwin Data Modeler упрощает разработку базы данных и автоматизирует множество трудоёмких задач, уменьшая сроки создания высококачественных и высокопроизводительных транзакционных баз данных и хранилищ данных.

В процессе информационного моделирования были выделены следующие сущности:

* Пользователь;
* Требование;
* Проект;
* Коллега,
* Заказчик.

Сущность «Требование» содержит информацию о типе, содержании, дате получения, дате выполнения, к какому проекту относится. Сущность «Требование» связана с сущностью «Проект» по названию проекта. Ключевым полем данной сущности является «Номер требования».

Сущность «Проект» содержит информацию о названии проекта, его кратком описании, руководителю. Ключевым полем данной сущности является «Название проекта»

Сущность «Коллега» содержит информацию об имени, фамилии, должности. Сущность «Коллега» связана с сущностью «Проект» по названию проекта. Ключевым полем данной сущности является номер коллеги.

Сущность «Заказчик» содержит информацию о наименовании заказчика. Сущность «Заказчик» связана с сущностью «Проект» по названию проекта. Ключевым полем данной сущности является номер заказчика.

Сущность «Пользователь» содержит информацию о логине, пароле, статусе пользователя. Ключевым полем данной сущности является номер пользователя.

С учетом выявленных связей между сущностями смоделируем их взаимодействие в формате IDEF1. Информационная модель соответствует условиям третьей нормальной формы – неключевые атрибуты зависят от ключей, причем от ключей целиком и ни от чего другого, кроме как от ключей.

Общая схема связанных сущностей базы данных представлена на рисунке 4.1.



Рисунок 4.1– Информационная модель системы

# **МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ**

Для описания представления системы используется язык UML.

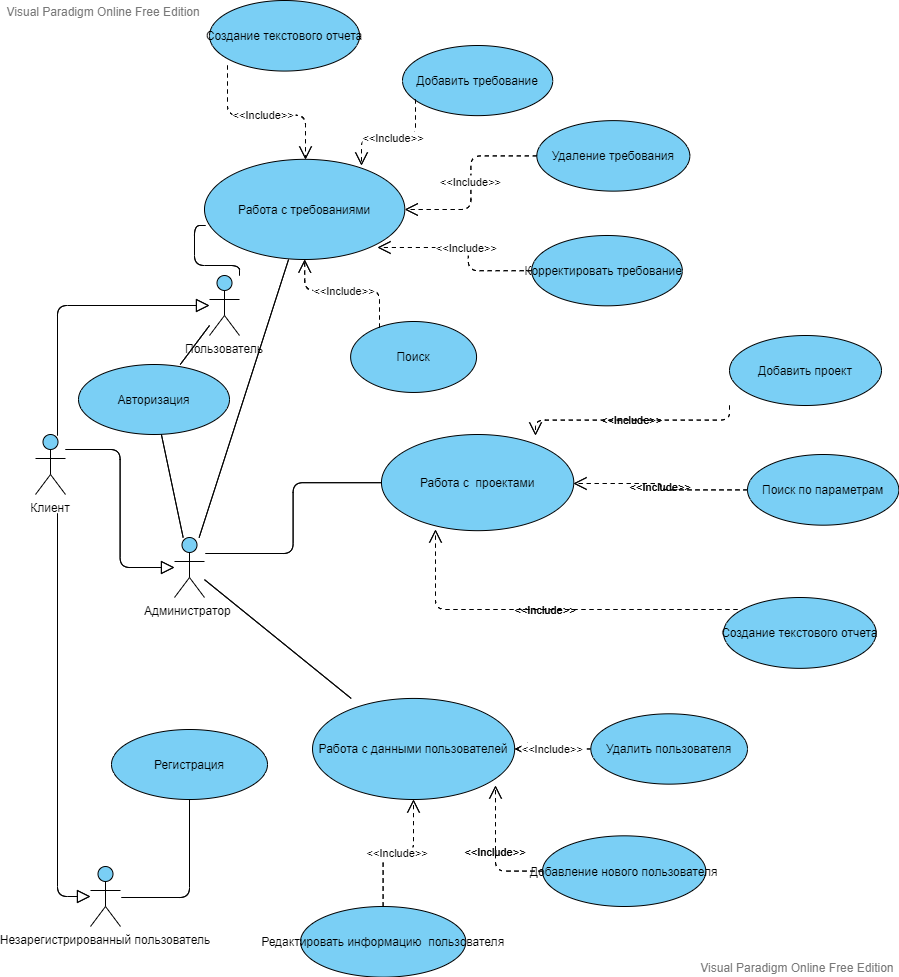
UML – [язык](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [графического](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) описания для [объектного моделирования](https://ru.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1) в области [разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F), [моделирования бизнес-процессов](https://ru.wikipedia.org/w/index.php?title=%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5&action=edit&redlink=1), [системного проектирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и отображения [организационных структур](https://ru.wikipedia.org/wiki/%D0%9E%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0).

UML является языком широкого профиля, использующий графические обозначения для создания [абстрактной модели](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), называемой UML-моделью.

Диаграммы UML используются для определения, визуализации, проектирования и документирования программных систем .

## **Диаграмма вариантов использования**

Диаграмма вариантов использования обеспечивает высокоуровневое описание того, что система в состоянии сделать и с кем (или чем) она будет взаимодействовать. На рисунке 5.1.1 представлена диаграмма вариантов использования для кассира и пользователя.



На рисунке 5.1.1 представлена диаграмма вариантов использования для пользователя, администратора и незарегистрированного пользователя.

## **Диаграмма состояний**

Диаграмма состояний описывает все возможные состояния одного экземпляра определенного класса и возможные последовательности его переходов из одного состояния в другое, то есть моделирует все изменения состояний объекта как его реакцию на внешние воздействия.

На рисунке 5.2.1 представлена диаграмма состояний авторизации.

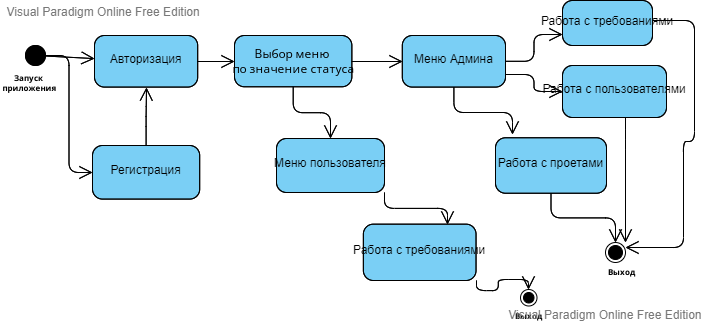


Рисунок 5.2.1 – Диаграмма состояний авторизации

## **Диаграмма последовательностей**

Диаграмма последовательности описывает поведение только одного варианта использования. На такой диаграмме отображаются только экземпляры объектов и сообщения, которыми они обмениваются между собой.

На рисунке 5.3.1 представлена диаграмма деятельности авторизации пользователя.

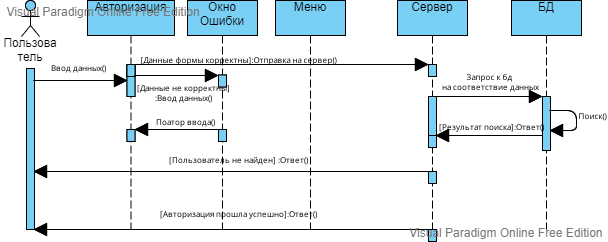
****

Рисунок 5.3.1 – Диаграмма последовательности авторизации пользователя

## **Диаграмма классов**

Диаграмма классов описывает типы объектов системы и различные отношения между ними.

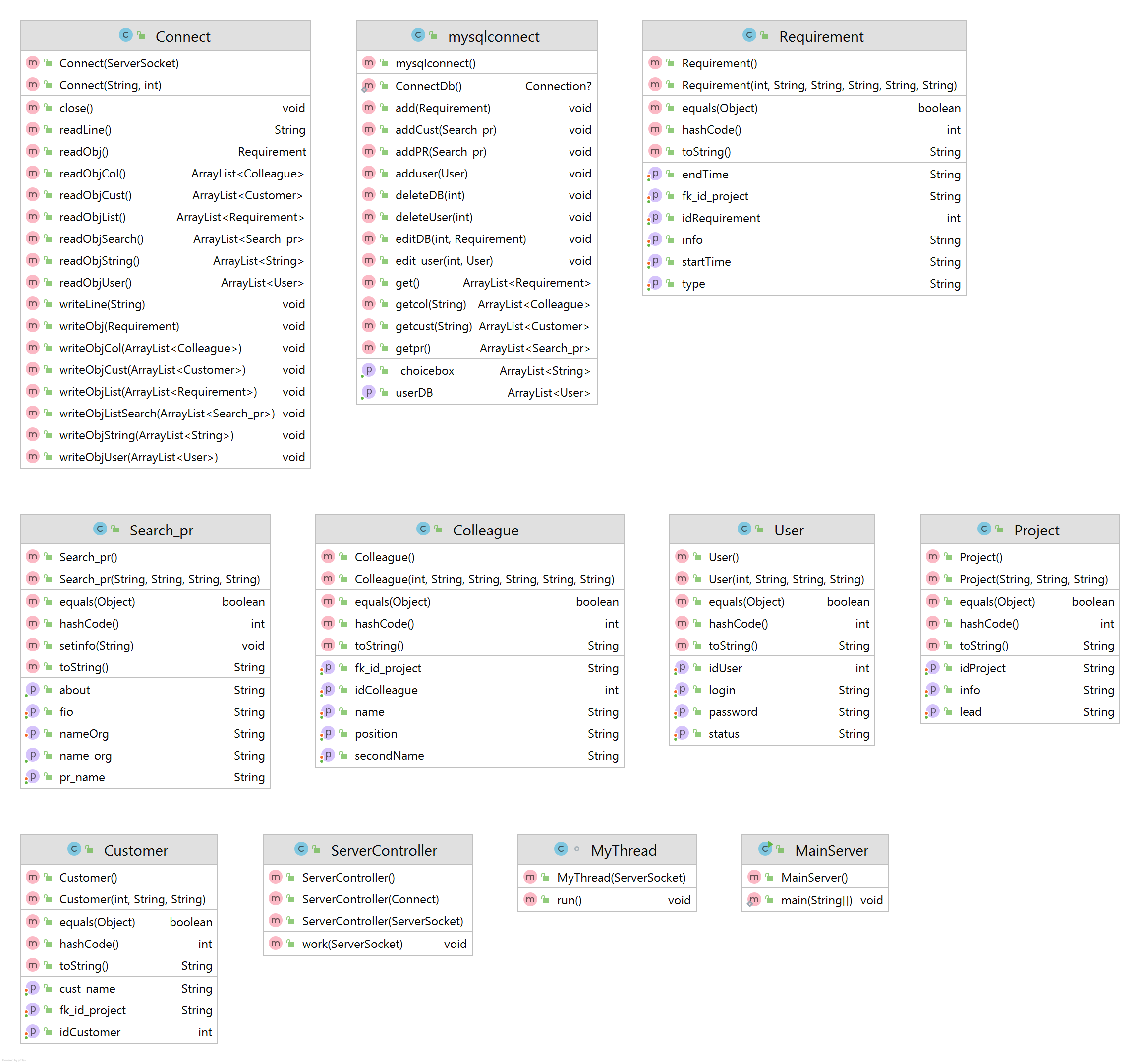


Рисунок 5.4.3 – Диаграмма классов, относящаяся к серверу

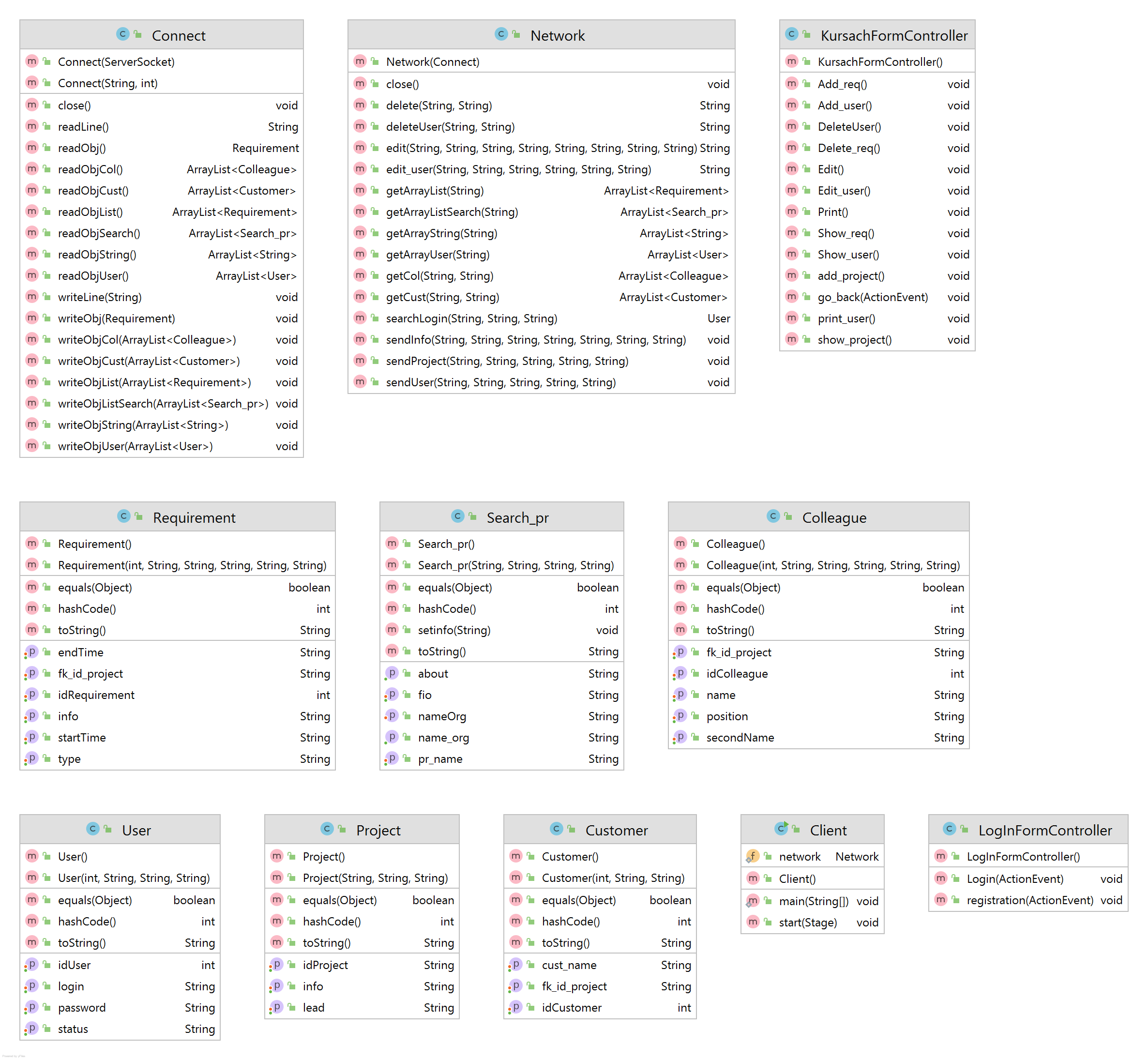


Рисунок 5.4.3 – Диаграмма классов , относящаяся к клиенту

## 

## **5.5 Диаграмма развертывания и компонентов**

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих лишь на этапе ее исполнения. При этом представляются только компоненты-экземпляры программы, являющиеся исполняемыми файлами или динамическими библиотеками.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними.

Диаграмма компонентов описывает особенности физического представления системы. Она позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный и исполняемый код.

На рисунке 5.5.1 представлена диаграмма развертывания и компонентов системы.

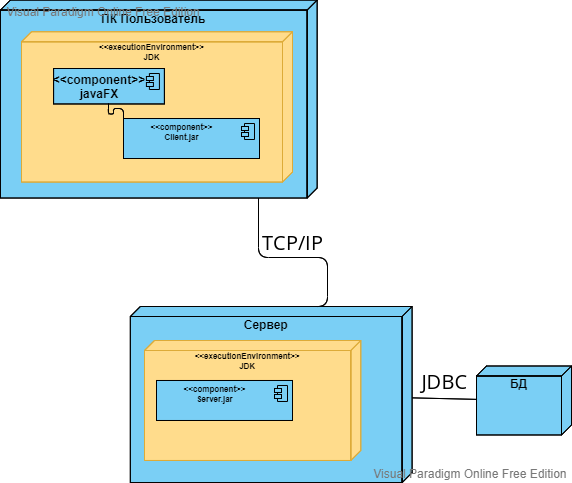


Рисунок 5.5.1 – Диаграмма развертывания и компонентов системы

# **6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ**

**6.1 Алгоритм удаления требования**

Рассмотрим алгоритм удаления требования. После того как пользователь вводит данные в текстовые поля, они отправляются на сервер. Сервер отправляет запрос к БД для того, чтобы найти соответствующую запись, если же запись существует сервер отправляет запрос на удаление, если же нет пользователь получит сообщение о том, что такой записи о требовании нет.

Блок-схема данного алгоритма представлена на рисунке 7.1.1.

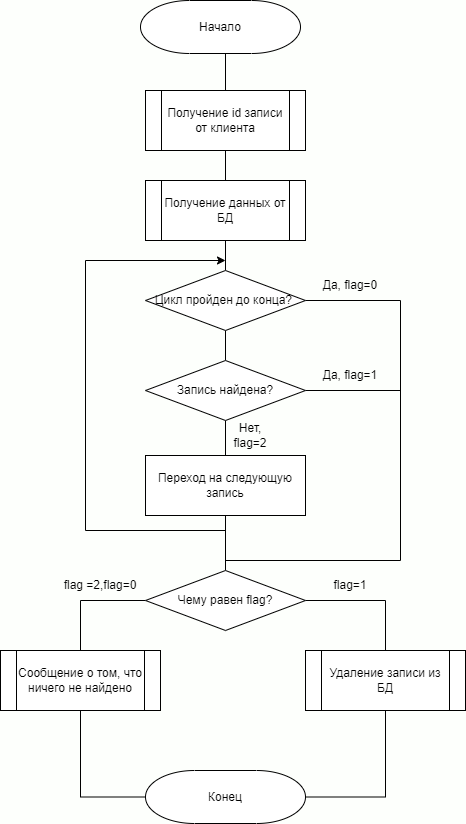


Рисунок 7.1.1 – Блок-схема алгоритма удаления требования

**6.2 Алгоритм добавления требования**

Пользователь заполняет текстовые поля после чего они отправляются на сервер. Сервер делает соответствующий запрос в БД, если записи с таким номером нет ее добавляют, в противоположном случае пользователь получает сообщение об ошибке.

Блок-схема данного алгоритма представлена на рисунке 7.2.1.

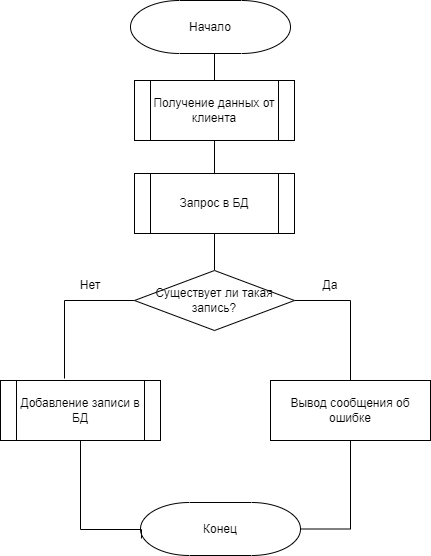


Рисунок 7.2.1 – Блок-схема алгоритма удаления требования

# **7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО РАЗВЕРТЫВАНИЮ СИСТЕМЫ**

Чтобы развернуть систему для сервера, необходимо установить MySQL Workbench 8.0. После установки открыть базу данных и запустить готовый скрипт. Так же необходимо наличие JavaVirtualMachine. Для запуска сервера нужно запустить файл Server.jar. При этом порт 1025 должен быть свободным, так как он установлен в курсовом проекте для соединения клиента с сервером.

После запуска этого файла будет отображена консоль, на которой будут появляться подключения новых клиентов, а также данные, которые будут передаваться либо получаться.

Скрипт генерации базы данных:

CREATE TABLE `colleague` (

`idcolleague` int NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`surname` varchar(45) NOT NULL,

`position` varchar(45) NOT NULL,

`fk\_id\_pr` varchar(45) NOT NULL,

PRIMARY KEY (`idcolleague`),

KEY `fk\_id\_pr\_idx` (`fk\_id\_pr`),

CONSTRAINT `fk\_id\_pr` FOREIGN KEY (`fk\_id\_pr`) REFERENCES `project` (`pr\_name`)

)

CREATE TABLE `customer` (

`id\_customer` int NOT NULL AUTO\_INCREMENT,

`name\_org` varchar(45) NOT NULL,

`fk\_cust\_pr` varchar(45) NOT NULL,

PRIMARY KEY (`id\_customer`),

KEY `fk\_id\_pr\_idx` (`fk\_cust\_pr`),

CONSTRAINT `fk\_cust\_pr` FOREIGN KEY (`fk\_cust\_pr`) REFERENCES `project` (`pr\_name`)

)

CREATE TABLE `project` (

`pr\_name` varchar(45) NOT NULL,

`lead` varchar(45) NOT NULL,

`about` varchar(100) DEFAULT NULL,

PRIMARY KEY (`pr\_name`)

)

CREATE TABLE `requirement` (

`id\_requirement` int NOT NULL,

`type` varchar(45) NOT NULL,

`info` varchar(200) NOT NULL,

`start\_time` varchar(45) NOT NULL,

`end\_time` varchar(45) NOT NULL,

`fk\_pr\_name` varchar(45) NOT NULL,

PRIMARY KEY (`id\_requirement`),

KEY `fk\_pr\_name\_idx` (`fk\_pr\_name`),

CONSTRAINT `fk\_pr\_name` FOREIGN KEY (`fk\_pr\_name`) REFERENCES `project` (`pr\_name`) ON DELETE RESTRICT ON UPDATE RESTRICT

)

CREATE TABLE `user` (

`iduser` int NOT NULL AUTO\_INCREMENT,

`login` varchar(45) NOT NULL,

`password` varchar(45) NOT NULL,

`status` varchar(45) NOT NULL,

PRIMARY KEY (`iduser`)

)

# **8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ**

После запуска клиентской части приложения вам будет предложено авторизироваться. Если же учетной записи нет, можно создать новую с помощью кнопки «Регистрация». Появится окно, представленное на рисунке 8.1

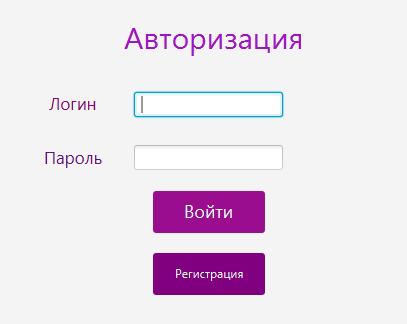


Рисунок 8.1 – Авторизация пользователя

Еслиавторизация прошла неуспешно вы увидите это сообщение. Представленное на рисунке 8.2

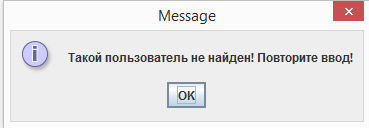


Рисунок 8.2 – Сообщение об ошибке

Если все же такая учетная запись существует вы увидите это сообщение, представленное на рисунке 8.3

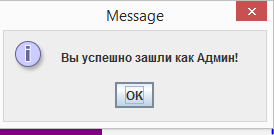


Рисунок 8.3 – Успешная вторизация

Если вы зашли в систему под статусом админа ваше меню выглядеть таким образом как на рисунке 8.4

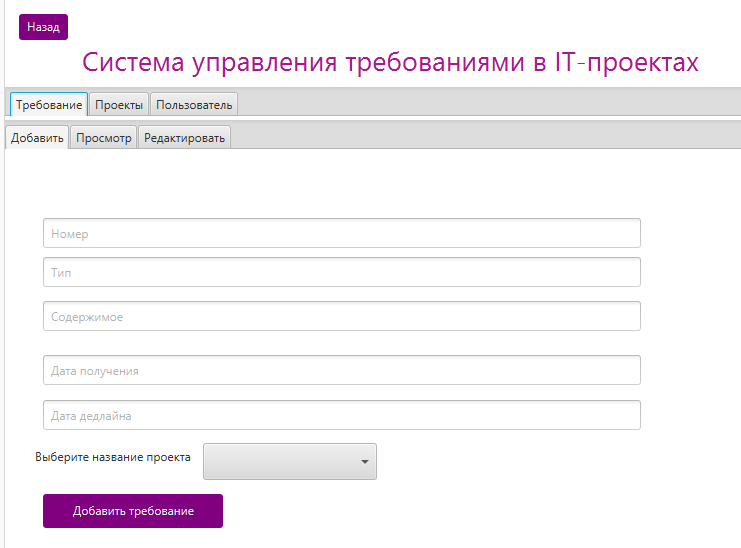


Рисунок 8.4 – Меню Администратора

Администратор может использовать весь функционал системы. Рассмотрим добавление нового требования: необходимо добавить корректные данные во все поля и начать кнопку «Добавить требование». Процесс добавления можно увидеть на рисунке 8.5

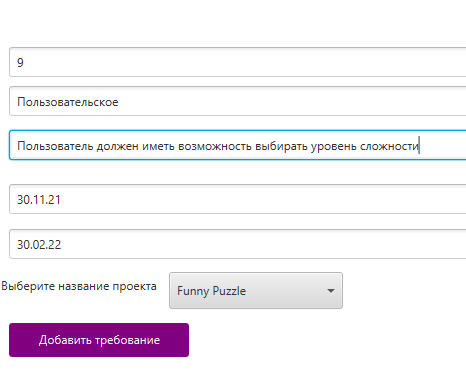


Рисунок 8.5 – Добавление требования

Результат успешно выполненной команды на рисунке 8.6:

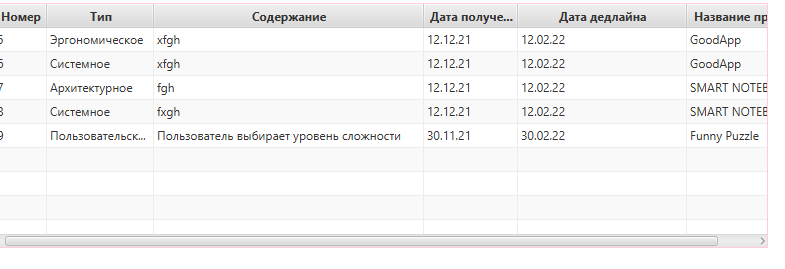


Рисунок 8.6 – Успешное добавление

Для того, чтобы удалить требование необходимо нажать на его в таблице или написать его номер в текстовое поле и нажать «Удалить» как представлено на рисунке 8.7:

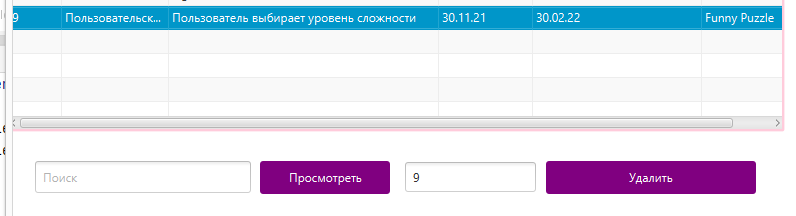


Рисунок 8.7 – Пример удаления

Для того, чтобы изменить данные о требовании необходимо заполнить все поля и нажать кнопку «Редактировать». В примере изменим Пользовательский тип требования на Архитектурный как представлено на рисунке 8.8:

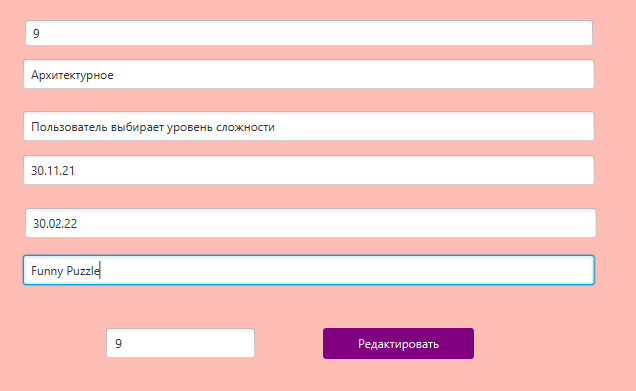


Рисунок 8.8 – Пример редактирования

Результат изменения данных о требовании:



Рисунок 8.9 – Результат редактирования

Для того чтобы найти нужное требование можно совершить поиск по Названию проекта, Типу и Содержанию требования. Для этого необходимо ввести текст в соответствующее поле:

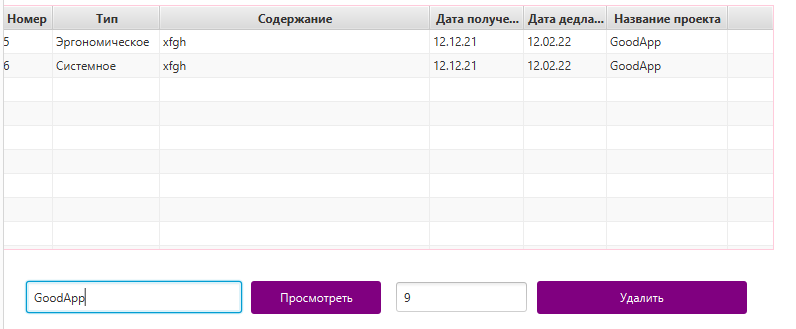


Рисунок 8.10 – Результат поиска по критерию

Следующая вкладка «Проект». Можно добавить,просмотреть, найти, распечатать информацию. Рассмотрим пример добавления проекта:

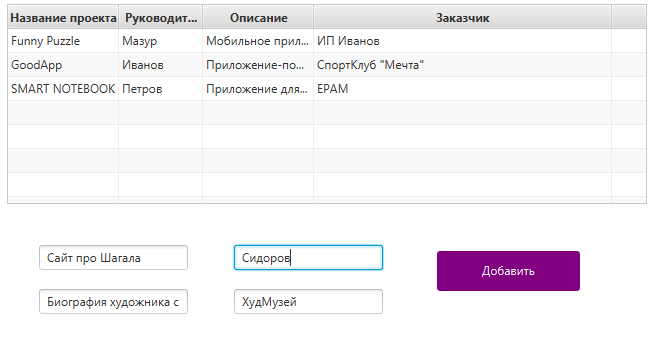


Рисунок 8.11 – Пример добавления проекта

Результат добавления представлен на рисунке 8.12:

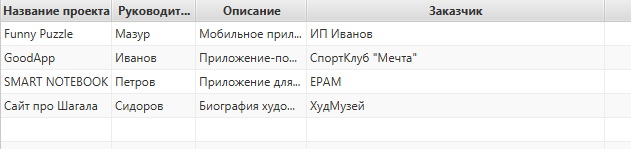


Рисунок 8.12 – Результат добавления

Вкладка «Пользователь». Есть возможность добавить, удалить, редактировать, распечатать информацию. Сама вкладка выглядит следующим образом, как представлено на рисунке 8.13:

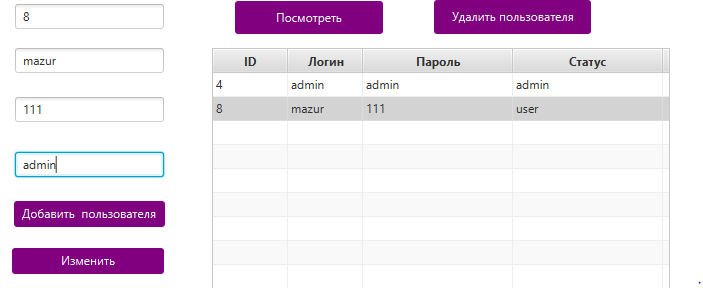


Рисунок 8.13 – Вкладка «Пользователь»

Есть возможность распечатать информацию(отчет) о проекте: требования, команда, заказчик. Для этого надо выбрать название проекта и нажать кнопку «Распечатать» как представлено на рисунке 8.14:



Рисунок 8.14 – Пример распечатки ответа

Обычный пользователь может использовать вкладку «Требование», но только с функцией просмотра, поиска и добавления.

# **ЗАКЛЮЧЕНИЕ**

На сегодняшний день, когда технологии не стоят на месте, а ведущие разработчики удивляют мир все новыми открытиями, очень важно идти в ногу с развитием.

По причине того, что с каждым годом появляются все новые разработки, названия которых многим и по сей день остаются незнакомыми, в мыслях многих людей возникает вопрос вроде: *«Автоматизация – что такое и для чего она необходима?»*

В виду того, что на улице уже 21 век, человечество давно решило перейти от работы, выполненной вручную, на использование машинной техники. Безусловно, ручной труд ценится и на сегодняшний день.Однако чтобы производительность была более продуктивной, одной такой работы будет мало.

Для того чтобы условия труда были максимально комфортными для людей, а процесс функционирования техники – более усовершенствованным и приспособленным для его легкой эксплуатации, технологами и была создана автоматизация.

На сегодняшний день **автоматизация процессов** в мире используется повсеместно – от координирования сложнейших производств до осуществления приобретений в супермаркетах. Направленность компании, равно как и ее масштабы, в данном случае не принципиальны: автоматика буквально пронизывает любую из них. А благодаря использованию процессорного подхода, для всей совокупности процессов применяются единые принципы автоматизации.

Что, в свою очередь, способствует: росту производительности;  
 ускорению; удешевлению; увеличению точности и стабильности.[8]

В случае системы управления требованиями на IT-проектах за счет использования базы данных, доступ любого сотрудника становится возможным, что повышает осведомленность и ответственность о стадии разработки продукта и о предоставленных требованиях для каждого члена команды. В результате работа становится более эффективной и качественной.

Таким образом, можно сделать вывод, что поставленные в начале работы цели и задачи были выполнены, программа соответствует необходимым нормам и готова к своему использованию на практике.

В дальнейшем приложение может быть усовершенствовано в соответствии с новыми идеями или требованиями.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] studref.com [Электронный ресурс]. – Электронные данные. – Режим доступа: https://studref.com/320281/informatika/razrabotka\_trebovaniy

[2] studme.org [Электронный ресурс]. – Электронные данные. – Режим доступа:https://studme.org/226098/informatika/klassifikatsiya\_trebovaniy\_vigersu

[3] Википедия. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/Разработка\_требовани й\_к\_ПО

[4] Методы определения требований в программной инженерии // НОУ «ИНТУИТ». [Электронный ресурс]. –– Электронные данные. –– Режим доступа: http://www.intuit.ru/studies/courses/2190/237/lecture/ 6122?page=1

[5] Википедия. [Электронный ресурс]. – Электронные данные. ––Режим доступа:https://ru.wikipedia.org/wiki/Управление\_требованиями\_к\_программному\_обеспечению

[6]. Системы управления требованиями: что и зачем? // Reqcenter. [Электронный ресурс]. –– Электронные данные. –– Режим доступа: http://edu.reqcenter.pro/?p=2433

[7] metanit.com [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/java/javafx/1.1.php>

[7] mosproject-eng.ru [Электронный ресурс]. – Электронные данные. – Режим доступа: https://mosproject-eng.ru/avtomatizacziya-proczessov.html

# 

# **ПРИЛОЖЕНИЕ А**

**(Обязательное)**