Министерство образования республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Институт информационных технологий

Специальность: Информационные системы и технологии (в обеспечении промышленной безопасности)

**Пояснительная записка**

**КУРСОВАЯ РАБОТА**

по курсу «Программирование сетевых приложений»

Тема: «Инструменты управления проектами в компании и программная поддержка количественной оценки потенциальных рисков IT-проекта»

Студент-вечерник 2 курса

группы № 980161

ФИО: Алейчик И.Д.

Минск, 2020

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

*(название ВУЗа)*

**Факультет:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«УТВЕРЖДАЮ»

**Заведующий кафедрой ЭИ**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020г.

**ЗАДАНИЕ**

**на курсовой проект**

**по дисциплине “Программирование сетевых приложений”**

**Студенту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_гр.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема проекта**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Срок сдачи студентом законченного проекта\_\_**\_\_\_\_\_\_\_15.12.2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3. Исходные данные к проекту**

**Общие требования.** Приложение должно быть выполнено в архитектуре клиент-сервер с многопоточным сервером с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

*Уровни архитектуры*: Серверное приложение может быть реализовано в виде консольного приложения или GUI-приложения. Настройки сервера должны меняться без изменения исходного кода (аргументы командной строки, конфиг-файлы и т.д.). Клиентское приложение: оконное приложение с использованием стандартных библиотек пользовательского интерфейса (JavaFX, SWING, AWT).

*В рамках работы над курсовым проектом должны быть использованы следующие техники*:

1. разработка и использование собственной иерархии классов (не менее 5), расширение базовых классов, предоставляемых JDK 7 и выше;
2. реализация не менее 2-х паттернов проектирования на свой выбор (по желанию).
3. использовать сокрытие данных (инкапсуляция), перегрузку методов, переопределение методов, сериализацию, абстрактные типы данных (интерфейсы, абстрактные классы), статические методы, обработку исключительных ситуаций.

**Бизнес-логика.** Бизнес-логика системы должна быть реализована только на серверной части приложения. На сервере должна быть предусмотрена возможность параллельной обработки запросов. Доступ к данным в СУБД должен осуществляться через драйвер, предоставляемый производителем СУБД или через использование специальных технологий.

*Обязательные требования к функционалу системы*:

1. Функционал приложения должен быть согласован с руководителем курсового проекта в срок согласно календарному плану работы над курсовым проектом. Диаграмма вариантов использования должна содержать ***не менее 15 Use Case,*** исключая тривиальные операции работы с БД (добавление, удаление, редактирование записей в БД).
2. В разрабатываемом приложении обеспечить добавление, редактирование и удаление записей из базы данных, сохранение табличных результатов в файле (создание текстового отчета), предоставление пользователю аналитической информации (графики, диаграммы).
3. Предусмотреть возможность сохранять информацию в любой момент на сервере и загружать ранее сохраненные данные.
4. В консольном окне серверной части должна отображаться текущая статистика подключений и параметры запуска сервера.
5. Предусмотреть механизм авторизации пользователей (роли Администратор и Пользователь).

База данных (не менее ***пяти!*** связанных таблиц) должна быть приведена к 3-ей нормальной форме.

**Транспортный уровень.** Взаимодействие между серверной и клиентскими частями должно осуществляться с использованием сокетов и протокола TCP/IP. Допускается использование технологии RMI.

**Требования к поставке.**

Разработанная система должна обладать следующей инфраструктурой:

- Исполняемые файлы должны работать в среде 32х разрядной ОС Windows 7 и выше

- СУБД (на выбор) – Sybase SQL 11.0+, MS SQL Server 2008 R2+, MySQL 5.5+, PostgreSQL 9.0+, Java DB 10.x+

База данных должна генерироваться sql-скриптом.

Интерфейс программы и данные должны быть только на русском языке.

Приложение должно запускаться без использования интегрированных средств разработки.

**При проектировании системы необходимо выполнить моделирование с использованием следующих стандартов:**

-IDEF0 – функциональное моделирование процессов предметной области решаемой задачи (не менее чем 4 уровня). Используется CASE средство allfusion process modeler.

-IDEF1.X – информационное моделирование (не менее 5 связанных сущностей) Используется CASE средство allfusion erwin data modeler

**-**UML 2.0 **–** модели представления системы на основе UML.

**Перечень используемых стандартов**:

ГОСТЫ 19001–19701 (оформление блок-схем, требования к руководству пользователя), стандарт оформления курсовых и дипломных работ БГУИР СТП 01-2017

**4. Содержание пояснительной записки**

Введение (с указанием цели и основных задач для её достижения). 1. Описание предметной области 2. Постановка задачи и обзор методов её решения. 3. Функциональное моделирование на основе стандарта IDEF0. Функциональная модель должна быть связана с информационной и cодержать декомпозиции бизнес-процессов с точки зрения предметной области. 4. Информационная модель системы и её описание (с доказательством приведения ее к 3-ей нормальной форме). 5. Обоснование оригинальных решений по использованию технических и программных средств, не включенных в требования (не обязательный раздел). 6. Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы. 7. Руководство пользователя. 8. Результаты тестирования разработанной системы. Заключение. Список использованных источников. Приложения.

**5. Перечень графического материала**

Функциональная модель процессов предметной области (IDEF0) – минимум 4 уровня. Информационная модель (IDEF1X). Диаграмма Вариантов использования (Use Case). Диаграммы состояний (Statechart). Диаграмма последовательностей (Sequence diagram). Диаграммы классов. Диаграммы компонентов (component diagram). Диаграмма развертывания (deployment diagram). Блок-схемы алгоритмов, реализующих бизнес-логику. Листинг алгоритмов, реализующих бизнес-логику. Листинг основных элементов программы. Листинг скрипта генерации базы данных.

**6. Дата выдачи задания**\_\_\_\_\_\_\_\_\_14.09.2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**7. Календарный график работы над проектом**

1.Функционирование контекстного уровня системы. Прототип пользовательского интерфейса. (15.10.2020)

2.Функционирование программного обеспечения, реализующего архитектуру системы. (15.11.2020)

3. Завершенная программная реализация системы (1012.2020).

4 Оформление пояснительной записки (15.12.2020)

**РУКОВОДИТЕЛЬ**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(ФИО, подпись)*

**Задание принял к исполнению**\_\_\_« 14 » сентября 2020г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(дата и подпись студента)*

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc59635392)

[1 Описание предметной облости 5](#_Toc59635393)

[1.1 Общие понятия и термины 5](#_Toc59635394)

[2 Постановка задачи и обзор методов ее решения 6](#_Toc59635395)

[3 Функциональная модель управления проектами и поддержки количественной оценки рисков 10](#_Toc59635396)

[4 Информационная модель системы и её описание 12](#_Toc59635397)

[5 Спецификация вариантов использования системы 15](#_Toc59635398)

[6 Модели представления системы и их описание 17](#_Toc59635399)

[6.1 Диаграммы состояний (Statechart diagram) 17](#_Toc59635400)

[6.2 Диаграмма последовательности регистрации нового пользователя (Sequence diagram) 17](#_Toc59635401)

[6.3 Диаграммы классов 18](#_Toc59635402)

[6.4 Диаграммы компонентов (component diagram) 18](#_Toc59635403)

[7 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы 19](#_Toc59635404)

[8 Руководство пользователя по развертыванию системы 20](#_Toc59635405)

[9 Результаты тестирования системы управления проектами в компании и программной поддержки количественной оценки потенциальных рисков IT-проекта 25](#_Toc59635406)

[Заключение 34](#_Toc59635407)

[Список использованных источников 35](#_Toc59635408)

[Приложение А(обязательное)Модели представления системы (к разделу 6) 36](#_Toc59635409)

[Приложение В(обязательное)Скрипт создания БД 40](#_Toc59635410)

[Приложение Г(обязательное)Выборочный листинг кода 45](#_Toc59635411)

# Введение

В настоящее время во всех отраслях человеческой деятельности огромное внимание уделяется использованию информационных технологий для наиболее рационального и эффективного использования ресурсов, имеющихся в распоряжении человека.

В современных условиях эффективное управление представляет собой ценный ресурс организации, наряду с финансовыми, материальными, человеческими и другими ресурсами. Следовательно, повышение эффективности становится одним из направлений совершенствования деятельности в целом. Наиболее очевидным способом повышения эффективности протекания трудового процесса является его автоматизация. Бурное развитие информационных компьютерных технологий, совершенствование технической платформы и появление принципиально новых классов программных продуктов привело к изменению подходов к автоматизации управления предприятиями, автоматизации документооборота, автоматизации различных процессов предприятий.

Сегодня актуальной является проблема эффективного управления рисками. Успех каждой организации зависит от ее способности адаптироваться на изменения внешнего окружения. Основу любой управленческой деятельности составляют принимаемые органами управления единолично или коллегиально, и которые направлены на достижение определенной цели, стоящей перед организацией. Одной из основных задач, которые выполняют в рамках управления ИТ-проектами, является управление рисками проектной деятельности, или управление рисками проекта. Это задача не отделяется от остальных функций управления ИТ-проектами.

При определении финансовых потребностей, вычисления сметы и бюджета, подготовка и заключение контрактов, при контроле за реализацией проекта стоит задача защиты участников проектной деятельности от различных видов рисков. Проект всегда нацелен на результат, на достижение определенных целей, на определенную предметную область. Реализация проекта осуществляется полномочным руководством проекта, менеджером проекта и командой проекта, работающей под этим руководством, другими участниками проекта, выполняющими отдельные специфические виды деятельности и процессы по проекту.

В работах по проекту, как правило, на условиях частичной занятости, могут участвовать представители линейных и функциональных подразделений компаний, ответственных за выполнение возложенных на них задач, видов деятельности, функций, включая планирование, руководство, контроль, организацию, администрирование и другие общесистемные функции

Таким образом, целью данного курсового проекта является сокращение временных затрат сотрудников, менеджеров за счёт автоматизации обработки необходимой информации: различных расчётов, процесса составления отчетности, представления расчётов в виде диаграмм.

Для достижения поставленной цели требуется решить следующие задачи:

* проанализировать заданную предметную область;
* рассмотреть физическую и логическую модель представления данных;
* создать базу данных;
* реализовать клиент с дружественным интерфейсом, обеспечивающий максимально удобное взаимодействие с программой или вычислительной системой.

# 1 Описание предметной облости

## 1.1 Общие понятия и термины

Риск — это неопределенное событие или условие, которое может повлиять на цели проекта. У рисков имеются свои источники и последствия. Риски отличаются от проблем и трудностей, так как имеют отношение к будущим, потенциально возможным негативным результатам. Проблемы же и трудности представляют собой нечто, имеющее место в настоящее время. Риски могут стать проблемами, если ими эффективно не управлять. При реализации любого ИТ-проекта особое внимание требуется уделять разработке и применению методов управления рисками.

Для управления рисками в проектах используются процедуры идентификации рисков, их качественная и количественная оценка. Еще на этапе формирования плана проекта осуществляется прогнозирование ситуаций, чтобы предупредить риски и разработать план реагирования на них.

Причиной возникновения рисков являются неопределенности, существующие в каждом проекте.

Планирование управления рисками подразумевает процесс принятия решений по применению и планированию управления рисками для конкретного проекта. Этот процесс может включать в себя решения по организации, кадровому обеспечению процедур управления рисками проекта, выбор предпочтительной методологии, источников данных для идентификации риска, временной интервал для анализа ситуации. Важно спланировать управление рисками, адекватное как уровню и типу риска, так и важности проекта для организации

Количественная оценка рисков определяет вероятность возникновения рисков и влияние последствий рисков на проект, что помогает группе управления проектами верно принимать решения и избегать неопределенностей.

Количественная оценка рисков позволяет определять:

* вероятность достижения конечной цели проекта;
* степень воздействия риска на проект и объемы непредвиденных затрат и материалов, которые могут понадобиться;
* риски, требующие скорейшего реагирования и большего внимания, а также влияние их последствий на проект;
* фактические затраты, предполагаемые сроки окончания.

# 2 Постановка задачи и обзор методов ее решения

В данном курсовом проекте необходимо реализовать приложение для управления проектами, которое должно включать в себя:

- реализацию базовой авторизации, для отображения имени пользователей и разграничения данных по уровням доступа;

- реализацию системы ведения данных;

- реализацию системы генерации отчетов и вывода в файлы;

- реализацию удобного интерфейса.

Также при сознании курсового проекта были использованы следующие программные 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. Java может использоваться для создания двух типов программ: приложений и апплетов. Приложения Java могут быть непосредственно выполнены, используя интерпретатор Java. Апплет - небольшая программа, работающая с окнами, которые внедрены в страницу HTML.

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.

Как правило, программа обработки данных (клиентская часть) располагается на одном персональном компьютере, а с 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.

PostgreSql – 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 PostgreSql можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

PostgreSql 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).

PostgreSql 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.

PostgreSql 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.

# 3 Функциональная модель управления проектами и поддержки количественной оценки рисков

Целью данного курсового проекта является создание программного продукта, который автоматизирует процесс ведения проектов и количественной оценки рисков проекта, а также на основе данных рисков расчёта стоимости проектов. Для этого необходимо более детально изучить полный процесс управления проектами и рисками. Для этого была разработана модель по методологии IDEF0, контекстная диаграмма верхнего уровня которого представлена на рисунке 3.1.



Рисунок 3.1– Контекстная диаграмма верхнего уровня

В основном процессе можно выделить следующие подпроцессы:

* определение проектов;
* идентификацию рисков и их оценку;
* определение стоимости проекта;
* уточнение и расчёт, сроков и стоимости внедрения проекта.

Декомпозиция контекстной диаграммы верхнего уровня показана на рисунке 3.2.



Рисунок 3.2– Декомпозиция контекстной диаграммы верхнего уровня

Процесс начинается с определения проекта. На этом этапе менеджер заполняет всю необходимую для дальнейших расчётов информацию. Далее, чтобы определить сроки и новую стоимость проекта, необходимо провести некоторые вычисления.

# 4 Информационная модель системы и её описание

В процессе информационного моделирования были выделены следующие сущности:

* Проект;
* Пользователи;
* Риски;
* Задачи
* Роли
* Журнал активности
* Валюта
* Приоритет
* Состояния

Сущности «Пользователь» и «Роли» необходимы для реализации процесса авторизации. Сущность «Пользователь» содержит все личные данные пользователя включая логин и пароль, а сущность «Роли» определяет, какие права доступа к данным будут предоставлены пользователю в зависимости от значения статуса. Эти сущности связаны между собой связью один-ко-многим, так как сразу несколько пользователей могут обладать одной ролью.

Сущность «Журнал активности» необходим для ведения отчетности о входах пользователях. Данная сущность связана с сущностью «Пользователи» связью один ко многим, т.к пользователь может быть один а в жернале могут быть отмечены много раз.

Сущность «Проект» необходима для хранения информации о проектах, для которых необходимо будет произвести оценку рисков и определить план задач. Она хранит такие данные, как название проекта, первоначальную стоимость и установленные сроки ввода эксплуатацию.

Сущность Валюта связана с сущностью «Проекты» связью один-ко-многим, т.к определнная стоемость проекта может быть в разной валюте, а валюта может быть у многих проетов.

Сущность «Риски» необходима для хранения данных о рисках проектов.Она хранит описание риска и возможные потери кторые может понети компания. Эта сущность связана с сущностью «Проекты» связью один-ко-многим, так как у определенного проекта может быть несколько рисков.

Сущность «Задачи» необходима для хранения данных о задачах кторые нужно выполнить для достижения цели. Она хранит описание задачи, сроки, состояния и приоритет. Сущности «Приоритет» и «Состояния» связаны с сущностью «Задачи», связью один-ко-многим, т.к приоритеты и состояния могут быть у многих задач.

С учетом выявленных связей между сущностями смоделируем их взаимодействие в формате IDEF1. Информационная модель соответствует условиям третьей нормальной формы – не ключевые 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.

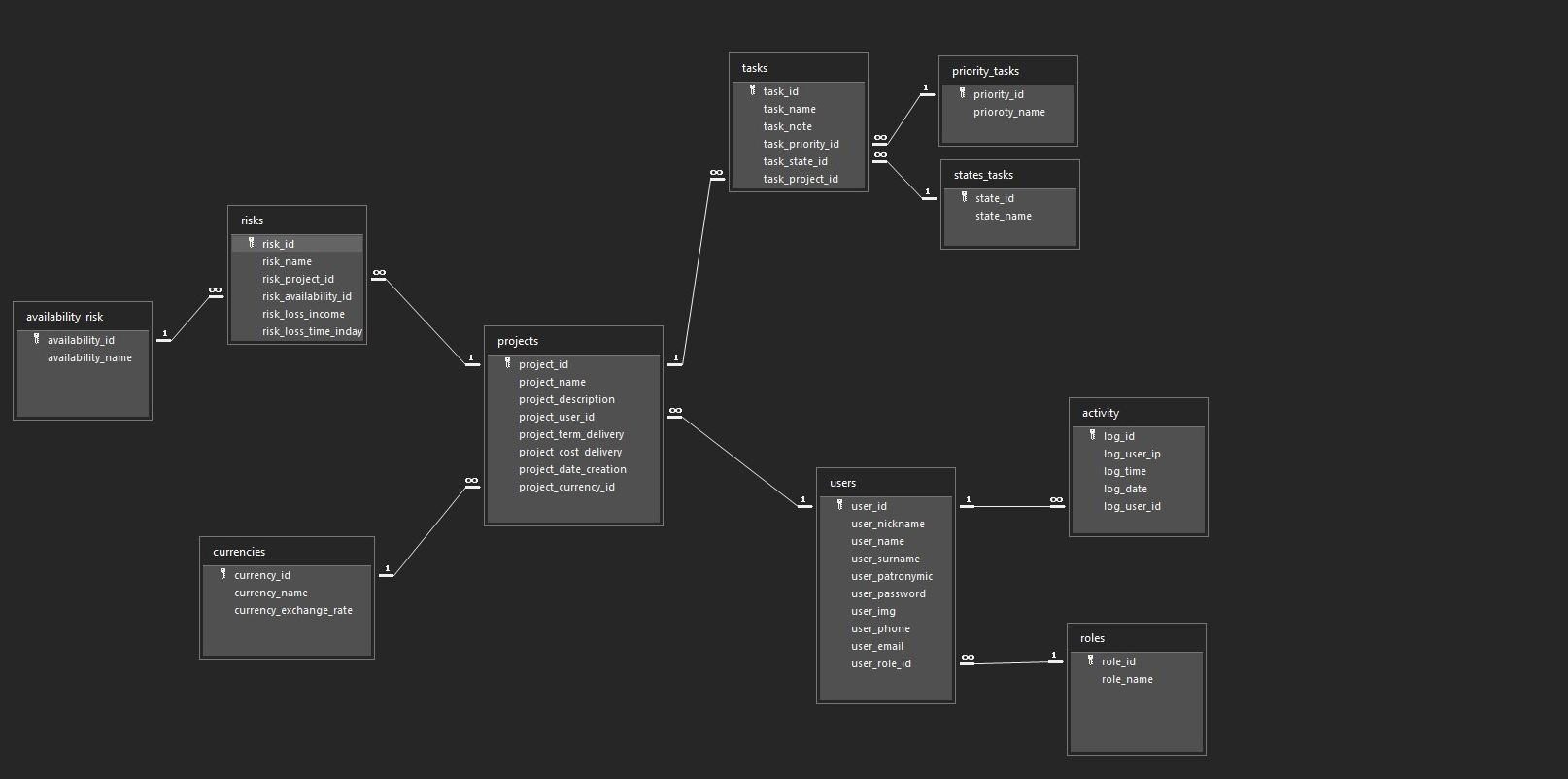


Рисунок 4.1– Информационная модель базы данных

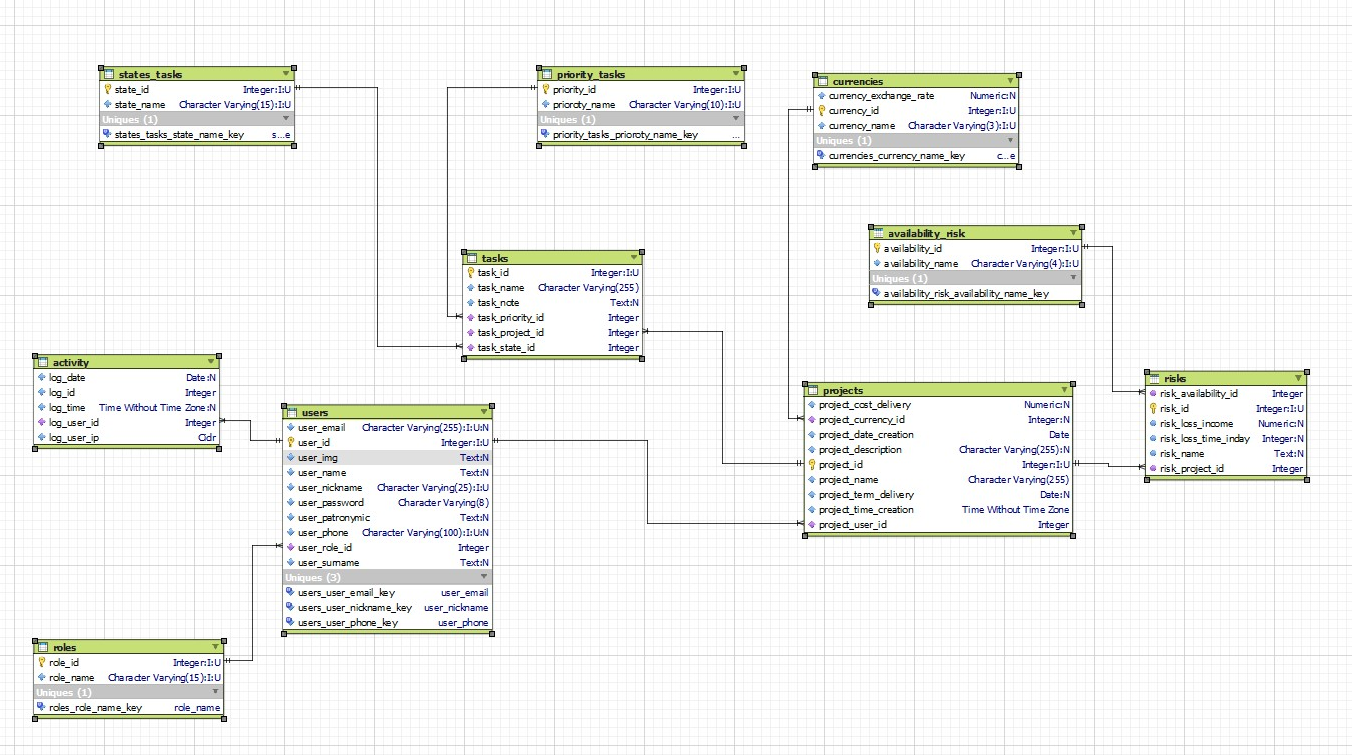


Рисунок 4.2– Информационная модель базы данных

# 5 Спецификация вариантов использования системы

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующих в процессе.

В данном курсовом проекте были определены четыре основных действующих лица: администратор, клиент, пользователь, незарегистрированный пользователь.

Последнему предоставлена единственная функция регистрации в системе. Пользователь же уже знает, что он зарегистрирован и он так же может получить доступ к системе с правами администратора.

Администратору и пользователю (менеджеру) представлены некоторые одинаковые функции, такие как:

* просмотр подробной информации об основных стоимостях проектов;
* поиск необходимой информации;
* сортировка по заданным полям;
* фильтрация;
* создание отчёта;
* выводы в файл.

Администратору предоставлен более широкий функционал, для него реализованы возможности:

* добавление новых пользователей (сотрудиков или администраторов);
* изменение статуса пользователей;
* редактирование имеющейся информации;
* удаление пользователей;
* просмотр журнала активности и его редактирование

При формировании отчета по данным проекта создается текстовый отчет с необходимой информацией.

Диаграмма вариантов использования представлена на рисунке 5.1.



Рисунок 5.1 – Диаграмма вариантов использования (Use Case)

# 6 Модели представления системы и их описание

## 6.1 Диаграммы состояний (Statechart diagram)

Диаграмма состояний определяют все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Рассмотрим диаграмму на примере работы данной программы (рисунок А.1).

В начале происходит запуск приложения. Далее происходит авторизация с возможностью зарегистрироваться как пользователь. В случае, если авторизация прошла успешно, пользователю предоставляется доступ к ресурсам, определяемый его статусом. У администратора, более широкий функционал. У него есть возможность работать с данными проектов, добавлять их в базу данных, а также работать с пользователями.

В любом из состояний, представленных на диаграмме, каждый пользователь имеет возможность завершить сеанс работы.

## 6.2 Диаграмма последовательности регистрации нового пользователя (Sequence diagram)

Диаграмма последовательностей относится к диаграммам взаимодействия UML, которые описывают поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательностей отображает временные особенности передачи и приема сообщений объектами. Диаграмма приведена на рисунке А.2.

Для того, чтобы добавить нового пользователя, необходимо сначала выбрать соответствующий пункт меню. Далее необходимо ввести данные аутентификации. После этого данные отправляются на сервер запросом, а сервер, в свою очередь, запрашивает данные о уже существующих пользователях в базе данных, сравнивает логин введённый новым пользователем с уже зарегистрированными. Если логины совпадают, пользователь не будет добавлен. Если же совпадений не было, то новый пользователь будет успешно добавлен и получит подтверждение от сервера.

## 6.3 Диаграммы классов

Так как это приложение создано с помощью клиент-серверной архитектуры, для клиента и для сервера будут определены свои отдельные диаграммы классов. Изображенные на рисунках A.3.1 – А.3.3.

## 6.4 Диаграммы компонентов (component diagram)

Диаграммы компонентов показывают, как выглядит модель системы на физическом уровне. На диаграмме изображены компоненты программного обеспечения и связи между ними.

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

Рассмотрим алгоритм расчёта стоимостей и сроков внедрения проектов на основе данных количественной оценки рисков, так как это является основной частью данного курсового проекта. Для того, чтобы сроки и стоимости внедрения, необходимы уже некоторые данные по данным проекта и его рискам в базе данных.

В клиентской части реализована функция, которая запрашивает данные об введённых уже рисках. Сервер получает эти данные с помощью класса ResultSet и передаёт их клиенту. Клиент, в свою очередь, получив данные, подставляет их в формулу расчета сроков и стоимости проекта. После чего отправляет рассчитанные данные серверу, а сервер добавляет новые записи в базу данных. Это основная бизнес-функция приложения.

Рассмотрим ещё один алгоритм: добавление нового пользователя которую может выполнить администратор.

Администратор нажимает кнопку добавления, после чего появляется панель регистрации, где необходимо ввести логин, а затем пароль. Если все данные корректно введены (пароль должен содержать не менее восьми символов), данные отправляются на сервер. Далее сервер запрашивает в базе данных выборку логинов уже зарегистрированных пользователей и сравнивает их с полученным. В случае, если совпадение произошло, сервер возвращает клиенту сообщение об ошибке. В противном случае, сервер продолжает проверку – сравнивает два последовательно введённых пароля на совпадение. Если пароли совпали – сервер отправляет клиенту сообщение об успешной регистрации, и делает запрос в базу данных о добавлении нового пользователя. Если же пароли не совпадают, клиент получит сообщение о несовпадении, которое отобразится на экране.

# 8 Руководство пользователя по развертыванию системы

Чтобы развернуть систему , необходимо установить PostgreSQL 13.0. После установки открыть базу данных и запустить готовый скрипт. Так же необходимо наличие JavaVirtualMachine. Для запуска программы нужно выполнить файл ProjectManager.jar..

После запуска этого файла будет отображен интерфейс, на котором нужно будет вывести все необходимые данные.

Скрипт для генерации базы данных:

DROP VIEW IF EXISTS projects\_view;

DROP VIEW IF EXISTS tasks\_view;

DROP VIEW IF EXISTS risks\_view;

DROP VIEW IF EXISTS activity\_log\_view;

DROP TABLE IF EXISTS activity,availability\_risk,currencies,priority\_tasks,projects,risks,roles,states\_tasks,tasks,users;

CREATE TABLE roles(

ROLE\_ID SERIAL PRIMARY KEY,

ROLE\_NAME CHARACTER VARYING( 15 ) UNIQUE NOT NULL

);

CREATE TABLE availability\_risk(

AVAILABILITY\_ID SERIAL PRIMARY KEY,

AVAILABILITY\_NAME CHARACTER VARYING( 4 ) UNIQUE NOT NULL

);

CREATE TABLE currencies(

CURRENCY\_ID SERIAL PRIMARY KEY,

CURRENCY\_EXCHANGE\_RATE NUMERIC,

CURRENCY\_NAME CHARACTER VARYING( 3 ) UNIQUE NOT NULL

);

CREATE TABLE states\_tasks(

STATE\_ID SERIAL PRIMARY KEY,

STATE\_NAME CHARACTER VARYING( 15 ) UNIQUE NOT NULL

);

CREATE TABLE priority\_tasks(

PRIORITY\_ID SERIAL PRIMARY KEY,

PRIOROTY\_NAME CHARACTER VARYING( 10 ) UNIQUE NOT NULL

);

CREATE TABLE users(

USER\_ID SERIAL PRIMARY KEY,

USER\_NAME TEXT,

USER\_SURNAME TEXT,

USER\_PATRONYMIC TEXT,

USER\_PASSWORD CHARACTER VARYING( 8 ) NOT NULL,

USER\_IMG TEXT,

USER\_PHONE CHARACTER VARYING( 100 ) UNIQUE NULL,

USER\_EMAIL CHARACTER VARYING( 255 ) UNIQUE NULL,

USER\_NICKNAME CHARACTER VARYING( 25 ) UNIQUE NOT NULL,

--FK-----------------------------------

USER\_ROLE\_ID INTEGER REFERENCES roles(ROLE\_ID) DEFAULT 3 NOT NULL

);

CREATE TABLE activity(

LOG\_ID SERIAL,

LOG\_TIME TIME,

LOG\_DATE DATE,

LOG\_USER\_IP CIDR NOT NULL,

--FK-----------------------------------

LOG\_USER\_ID INTEGER REFERENCES users(USER\_ID) NOT NULL

);

CREATE TABLE projects(

PROJECT\_ID SERIAL PRIMARY KEY,

PROJECT\_DATE\_CREATION DATE NOT NULL,

PROJECT\_TIME\_CREATION TIME NOT NULL,

PROJECT\_TERM\_DELIVERY DATE,

PROJECT\_COST\_DELIVERY NUMERIC,

PROJECT\_NAME CHARACTER VARYING( 255 ) NOT NULL,

PROJECT\_DESCRIPTION CHARACTER VARYING( 255 ),

--FK-----------------------------------

PROJECT\_CURRENCY\_ID INTEGER REFERENCES currencies(CURRENCY\_ID),

PROJECT\_USER\_ID INTEGER REFERENCES users(USER\_ID) NOT NULL

);

CREATE TABLE risks(

RISK\_ID SERIAL PRIMARY KEY,

RISK\_LOSS\_INCOME NUMERIC,

RISK\_LOSS\_TIME\_INDAY INTEGER,

RISK\_NAME TEXT,

--FK-----------------------------------

RISK\_AVAILABILITY\_ID INTEGER REFERENCES availability\_risk(AVAILABILITY\_ID) NOT NULL,

RISK\_PROJECT\_ID INTEGER REFERENCES projects(PROJECT\_ID) NOT NULL

);

CREATE TABLE tasks(

TASK\_ID SERIAL PRIMARY KEY,

TASK\_NAME CHARACTER VARYING( 255 ) NOT NULL,

TASK\_NOTE TEXT,

--FK-----------------------------------

TASK\_STATE\_ID INTEGER REFERENCES states\_tasks(STATE\_ID) NOT NULL,

TASK\_PRIORITY\_ID INTEGER REFERENCES priority\_tasks(PRIORITY\_ID) NOT NULL,

TASK\_PROJECT\_ID INTEGER REFERENCES projects(PROJECT\_ID) NOT NULL

);

INSERT INTO roles(ROLE\_NAME) VALUES('Администратор'),('Модератор'),('Пользователь');

INSERT INTO availability\_risk(AVAILABILITY\_NAME) VALUES('Есть'),('Нет');

INSERT INTO currencies(CURRENCY\_EXCHANGE\_RATE,CURRENCY\_NAME) VALUES(2.53,'USD'),(3.07,'EUR'),(3.44,'RUB'),(19.387,'BTC');

INSERT INTO states\_tasks(STATE\_NAME) VALUES('Инициирование'),('Планировние'),('Разработка'),('Одобрение'),('Закрыт');

INSERT INTO priority\_tasks(PRIOROTY\_NAME) VALUES('Высокий'),('Средний'),('Низкий');

CREATE VIEW projects\_view AS SELECT

projects.project\_id,

projects.project\_date\_creation,

projects.project\_time\_creation,

projects.project\_term\_delivery,

projects.project\_cost\_delivery,

projects.project\_name,

projects.project\_description,

currencies.currency\_exchange\_rate,

currencies.currency\_name,

projects.project\_user\_id

FROM projects

INNER JOIN currencies ON currencies.currency\_id = projects.project\_currency\_id ;

--RISKS VIEW

CREATE VIEW risks\_view AS SELECT

risk\_id,

risk\_name,

risk\_loss\_income,

risk\_loss\_time\_inday,

projects.project\_name,

availability\_risk.availability\_name,

availability\_risk.availability\_id,

projects.project\_id,

projects.project\_user\_id

FROM risks

INNER JOIN availability\_risk ON availability\_risk.availability\_id = risk\_availability\_id

INNER JOIN projects ON projects.project\_id = risk\_project\_id;

---TASKS VIEW

CREATE VIEW tasks\_view AS SELECT task\_id,

task\_name,

task\_note,

states\_tasks.state\_name,

priority\_tasks.prioroty\_name,

projects.project\_name,

states\_tasks.state\_id,

priority\_tasks.priority\_id,

projects.project\_user\_id,

projects.project\_id

FROM tasks

INNER JOIN states\_tasks ON states\_tasks.state\_id = task\_state\_id

INNER JOIN priority\_tasks ON priority\_tasks.priority\_id = task\_priority\_id

INNER JOIN projects ON projects.project\_id = task\_project\_id;

CREATE VIEW activity\_log\_view AS SELECT

log\_time,

log\_date,

log\_user\_ip,

users.user\_email,

users.user\_nickname,

roles.role\_name

FROM activity

INNER JOIN users ON users.user\_id = activity.log\_user\_id

INNER JOIN roles ON roles.role\_id = users.user\_role\_id;

# 9 Результаты тестирования системы управления проектами в компании и программной поддержки количественной оценки потенциальных рисков IT-проекта

После запуска исполняемого файла появится окно, представленное на рисунке 9.1.

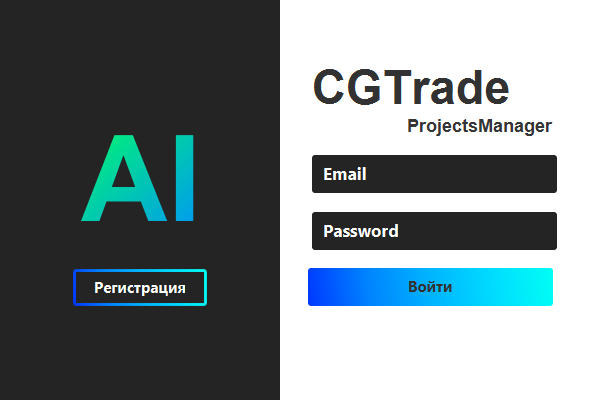


Рисунок 9.1 – Авторизация пользователя

Для того, чтобы успешно пройти авторизацию, необходимо ввести верные данные аутентификации. Система требует заполнения обеих полей. При заполнении неверными данными появится окно, представленное на рисунке 9.2.



Рисунок 9.2 – Ошибка при попытке авторизации

Далее введём корректные данные, чтобы рассмотреть функционал пользователя (рисунок 9.3)

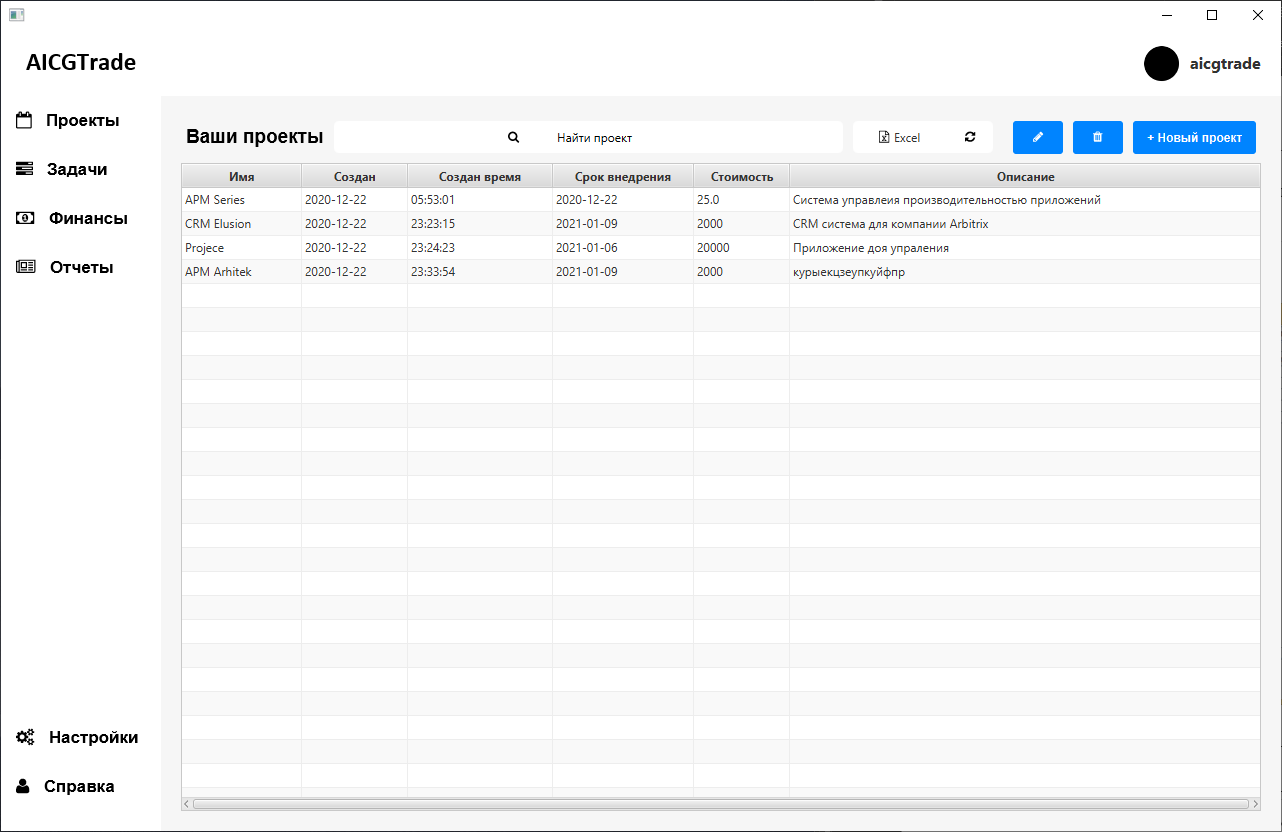


Рисунок 9.3 – Окно пользователя

На рисунке представлена первая таблица «Проекты». Рассмотрим добавление новой записи. При нажатии на кнопку «Новый проект», появляется окно следующего вида (рисунок 9.4).

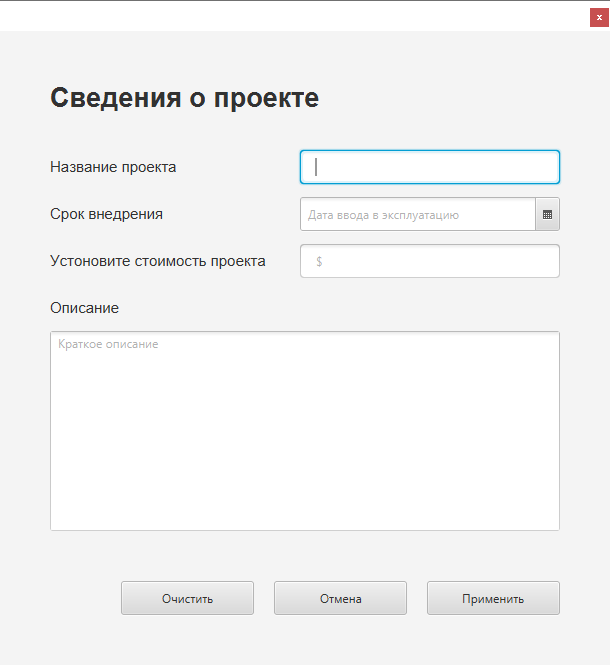


Рисунок 9.4 – Добавление новой записи

После нажатия на кнопку «Применить», появляется сообщение об успешном добавлении (рисунок 9.5).

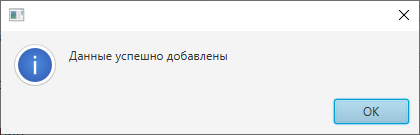


Рисунок 9.5 – Сообщение об успешном добавлении

Теперь таблица «Проекты» была обновлена, и можно убедиться, что данные действительно были добавлены (рисунок 9.6).

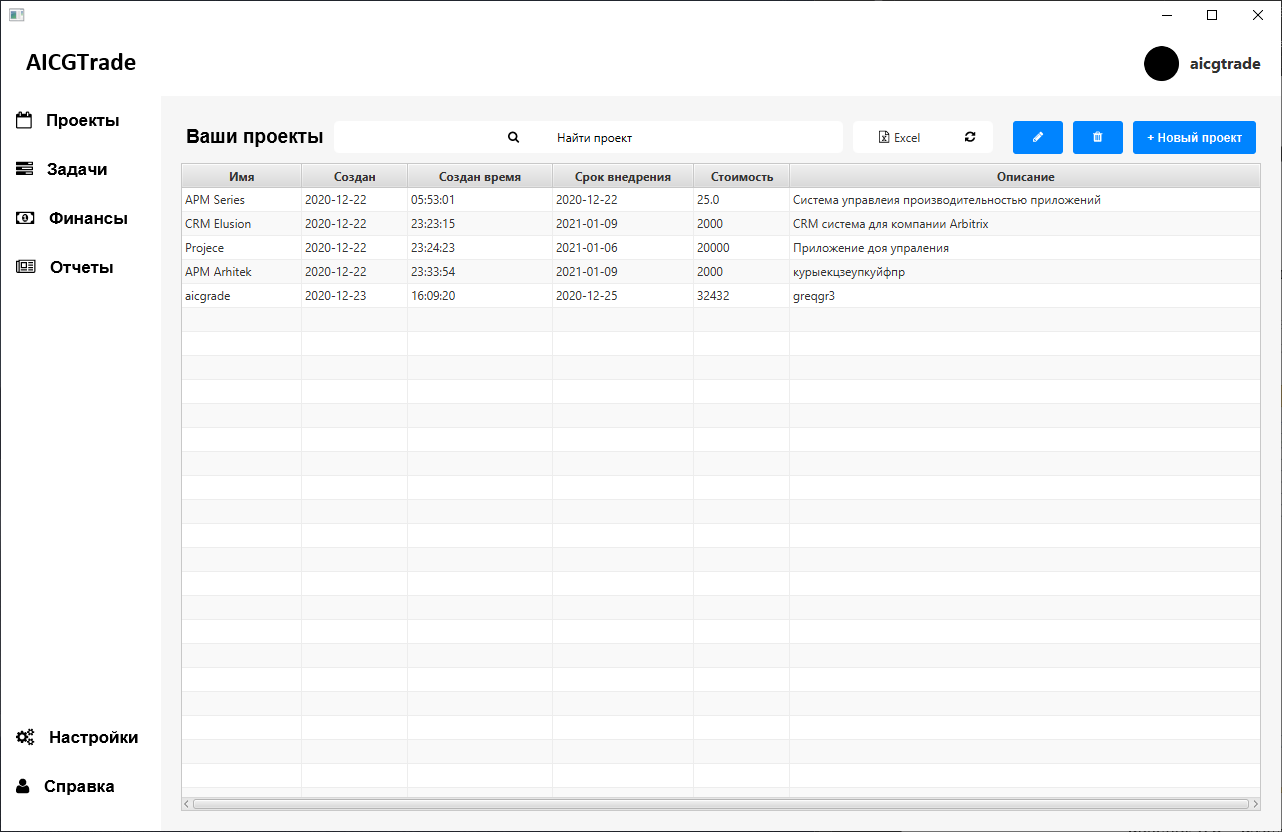


Рисунок 9.6 – Результат добавления новой записи

Рассмотрим удаление записи. Для этого следует обязательно выбрать одну запись, больше выбрать не получится (рисунок 9.7).

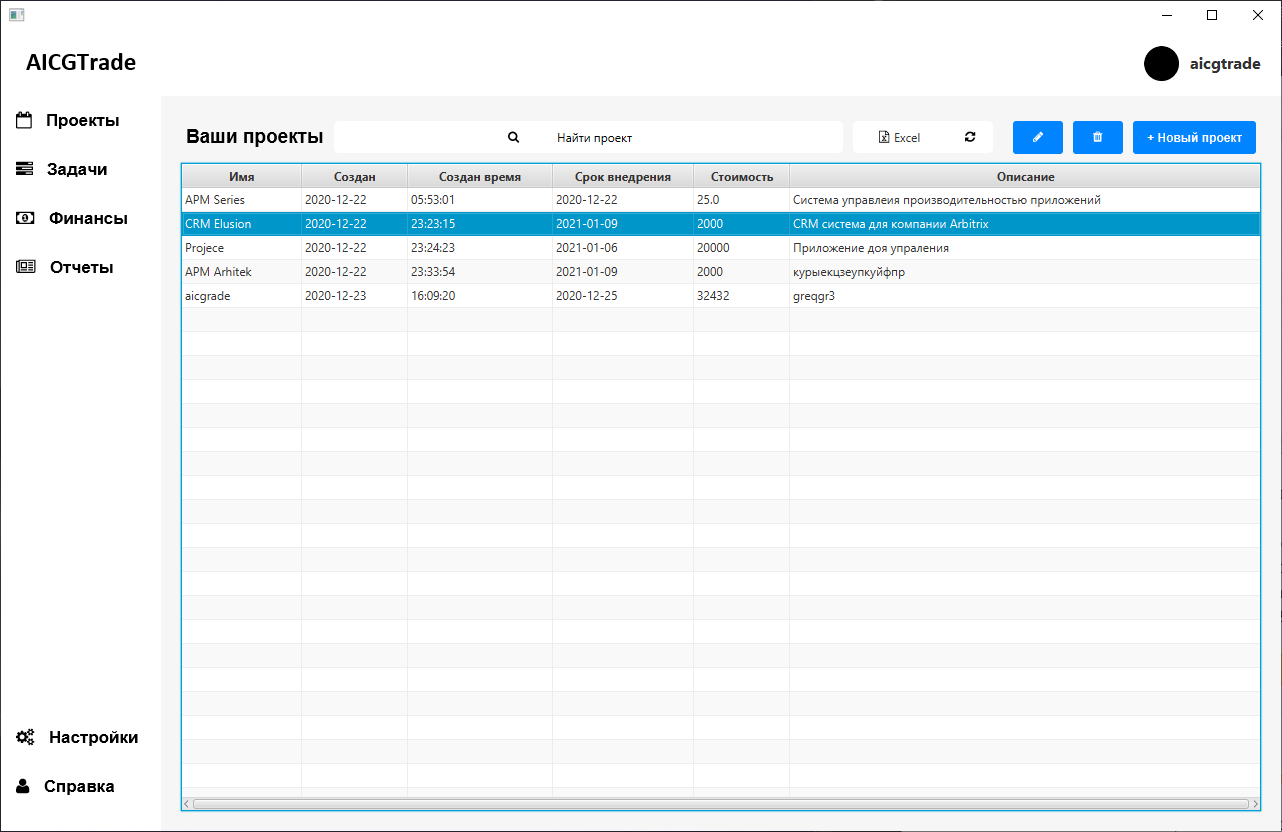


Рисунок 9.7 – Выбираем запись

После удаления, таблицу следует обновить, потом убедимся, что запись удалена (рисунок 9.8).

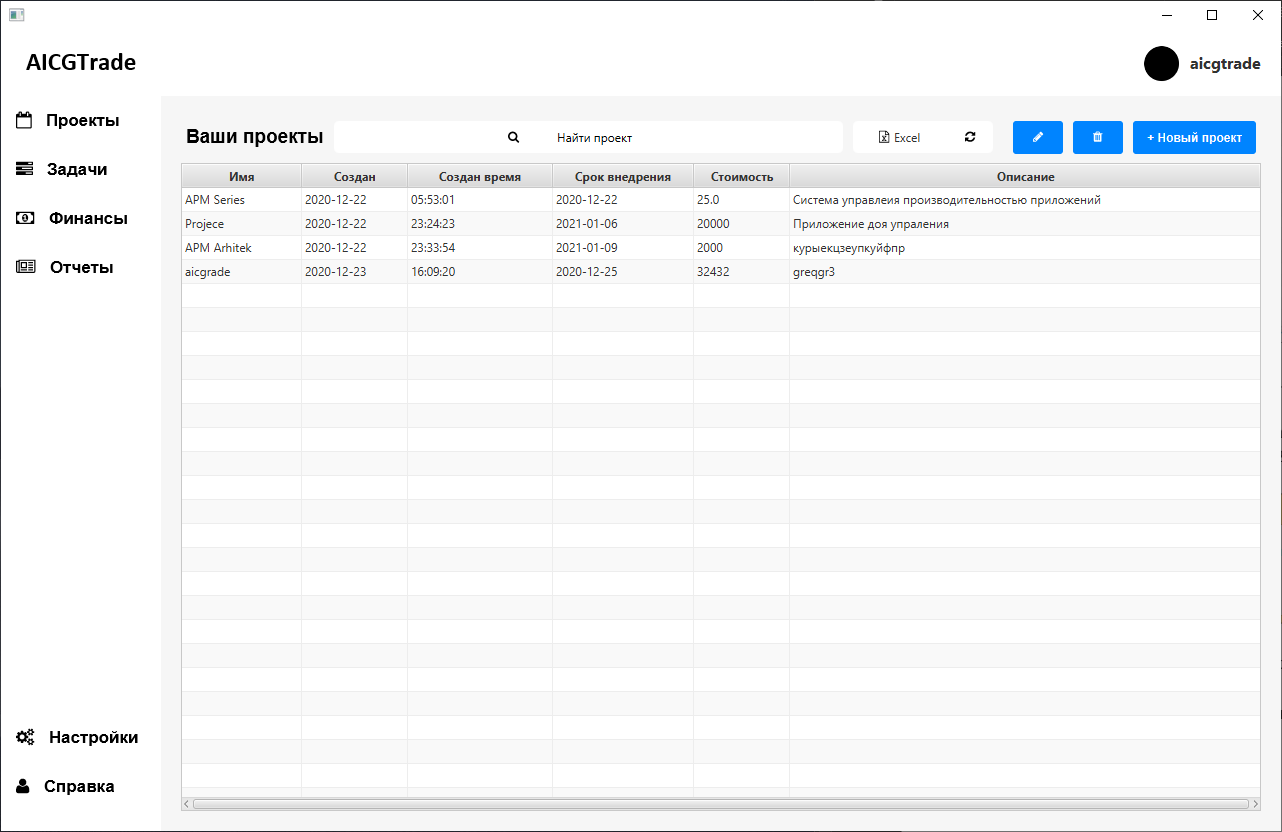


Рисунок 9.8 – Удаление записи

Далее перейдем на вкладку «Задачи». На рисунке 9.9 представлена таблица с задачами. Что бы отрисовать данные следует выбрать проект из выпадающего меню. Также на рисунке пользователю предоставляется возможность отсортировать данные по необходимым параметрам. Также построение таблицы и создание отчёта.

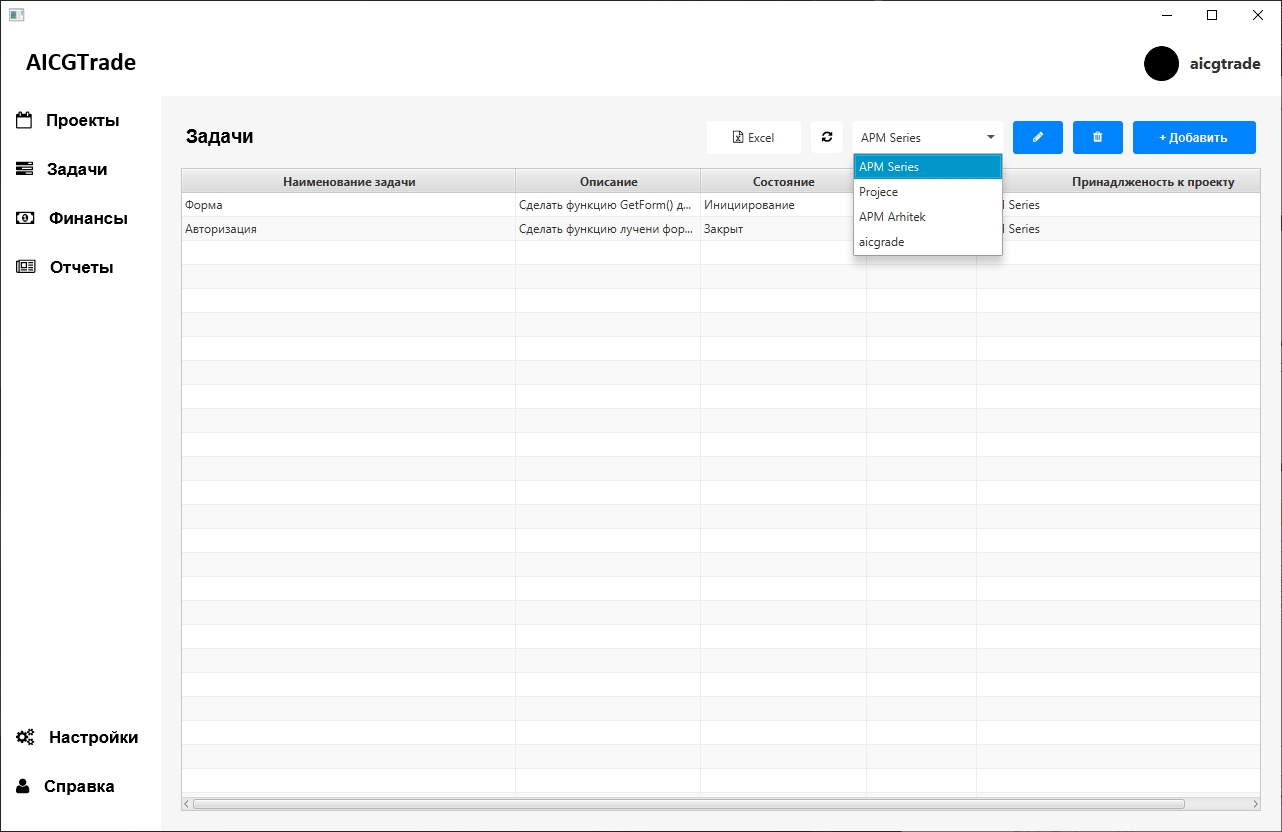


Рисунок 9.9 – Вкладка «Задачи»

Далее рассмотрим функцию поиска. В поле введем значение «Имя», а результат поиска представлен на рисунке 9.10.

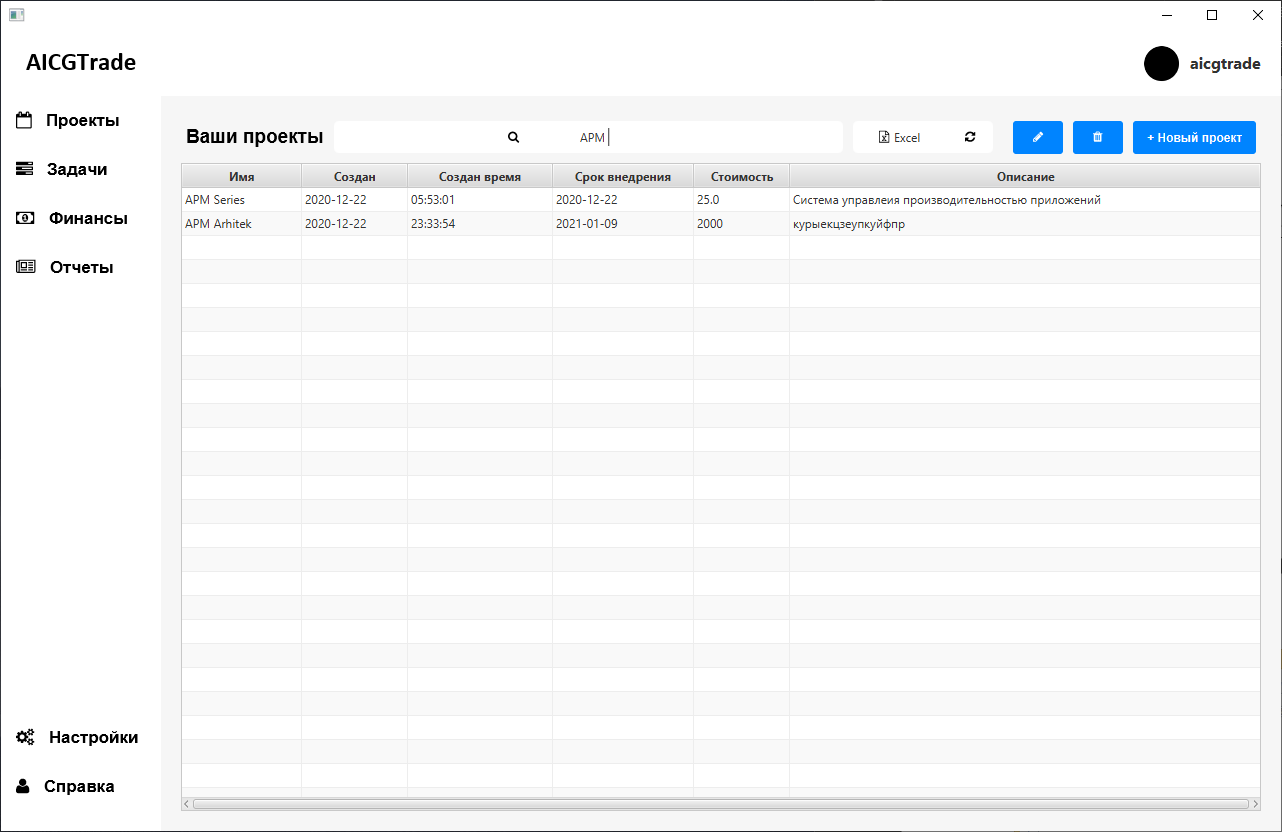


Рисунок 9.10 – Результат поиска

Перейдем на страницу «Финансы», в нижней части формы можно увидеть кнопку подробные данные о проекте. Выбрав в выпадающем списке проект, появится дополнительная информация по конкретному проекту из таблицы «Риски» (рисунок 9.11).

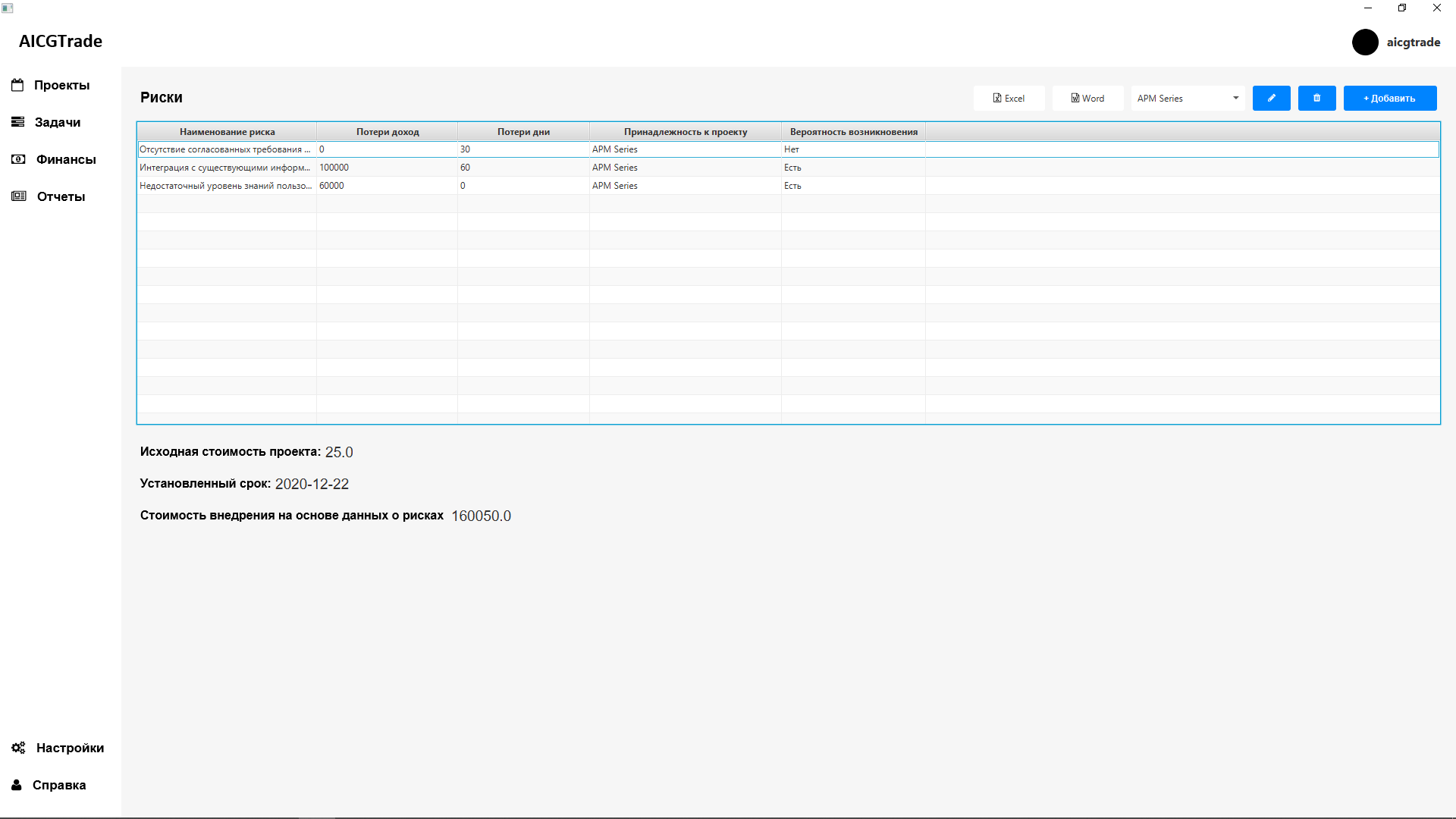


Рисунок 9.11 – Результат выбора проекта в таблице «Риски»

Далее попробуем составить отчёт. При нажатии на кнопку «Word» появится окно (рисунок 9.12), а отчёт представлен на рисунке 9.13.

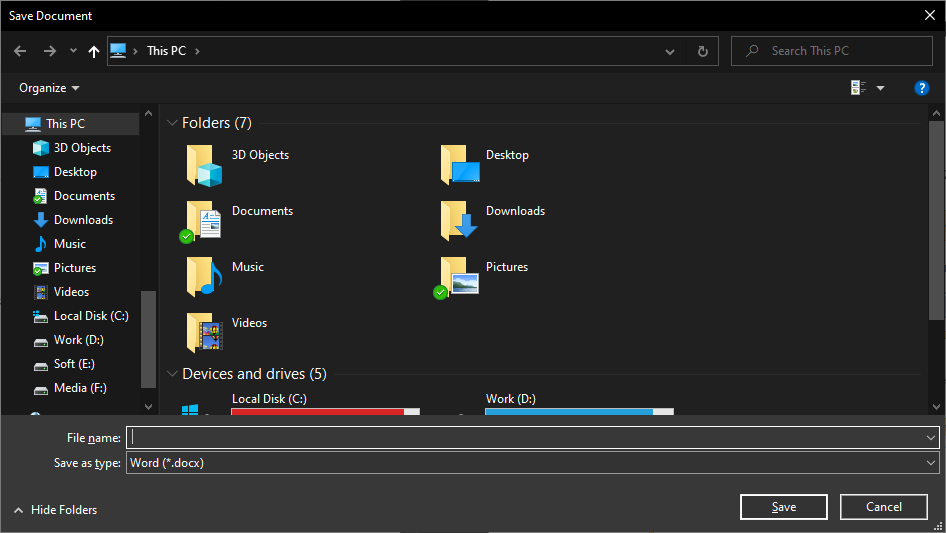


Рисунок 9.12 – Окно выбора пути сохранения отчёта

Далее нажимаем на кнопку «Сохранить» и переходим к отчету, по выбранному пути сохранения.

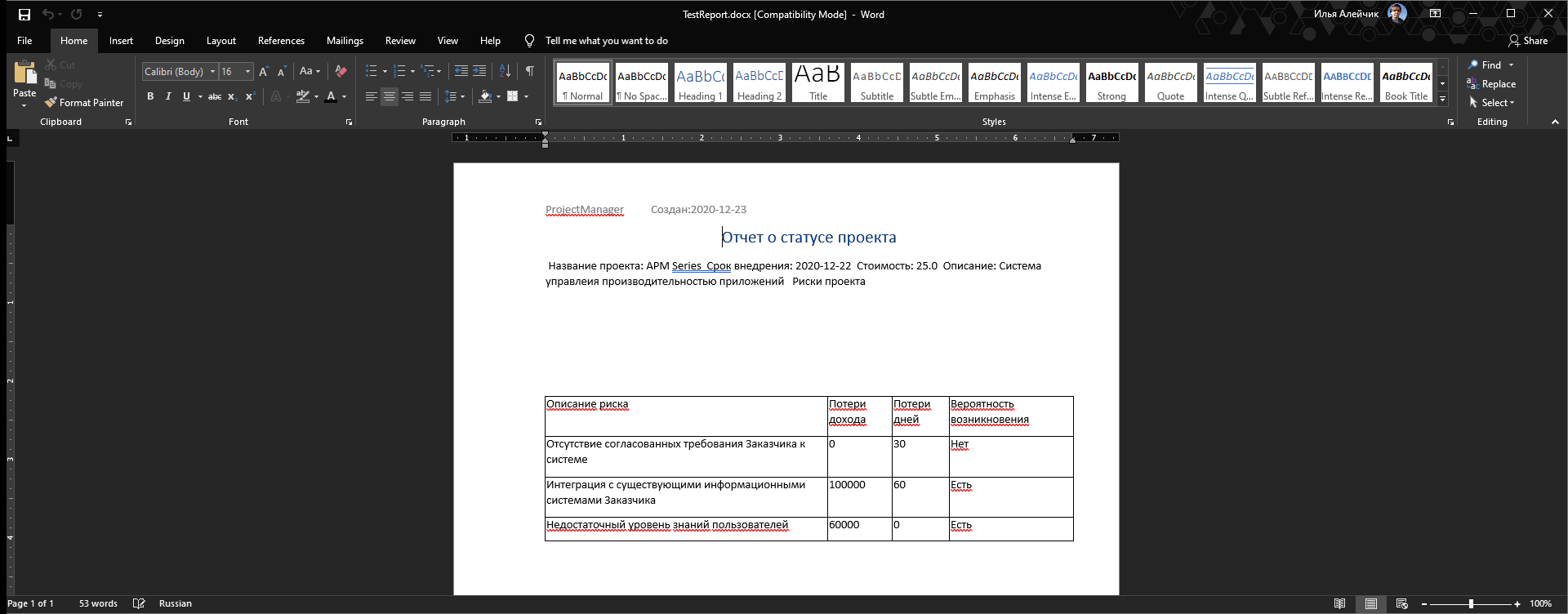


Рисунок 9.13 – Созданный отчёт

Далее перейдем на вкладку «Настройки», где сразу выводится таблица со всеми входами в систему текущего пользователя(рисунок 9.14). Так же на данной вкладки располагается все информация о программном обеспечении, ее версии и разработчике.

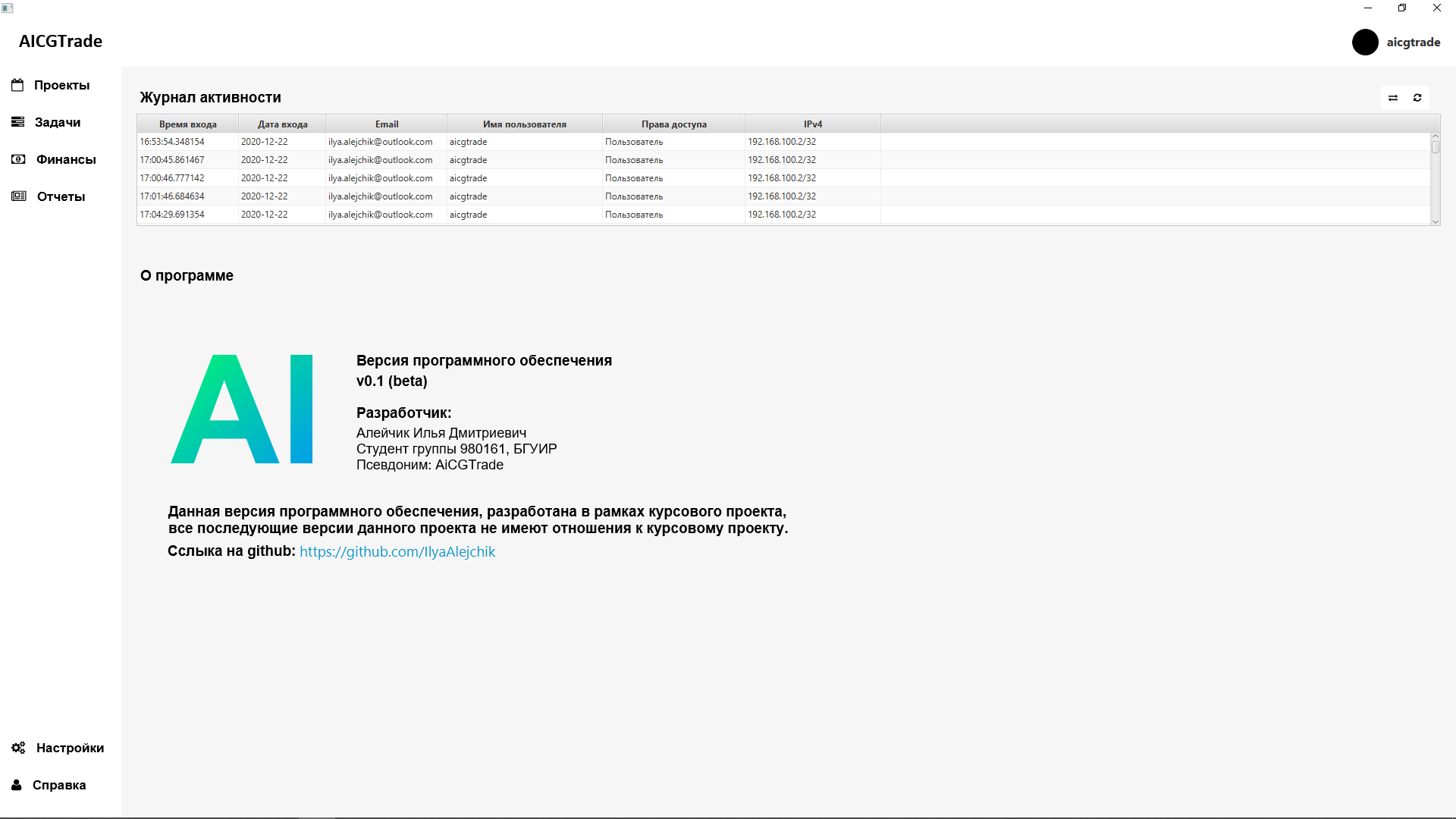


Рисунок 9.14 – Вкладка «Настройки»

Теперь перейдем на вкладку «Справка», где сразу выводится страница со всей информацией о программе и руководстве пользователя(рисунок 9.15).

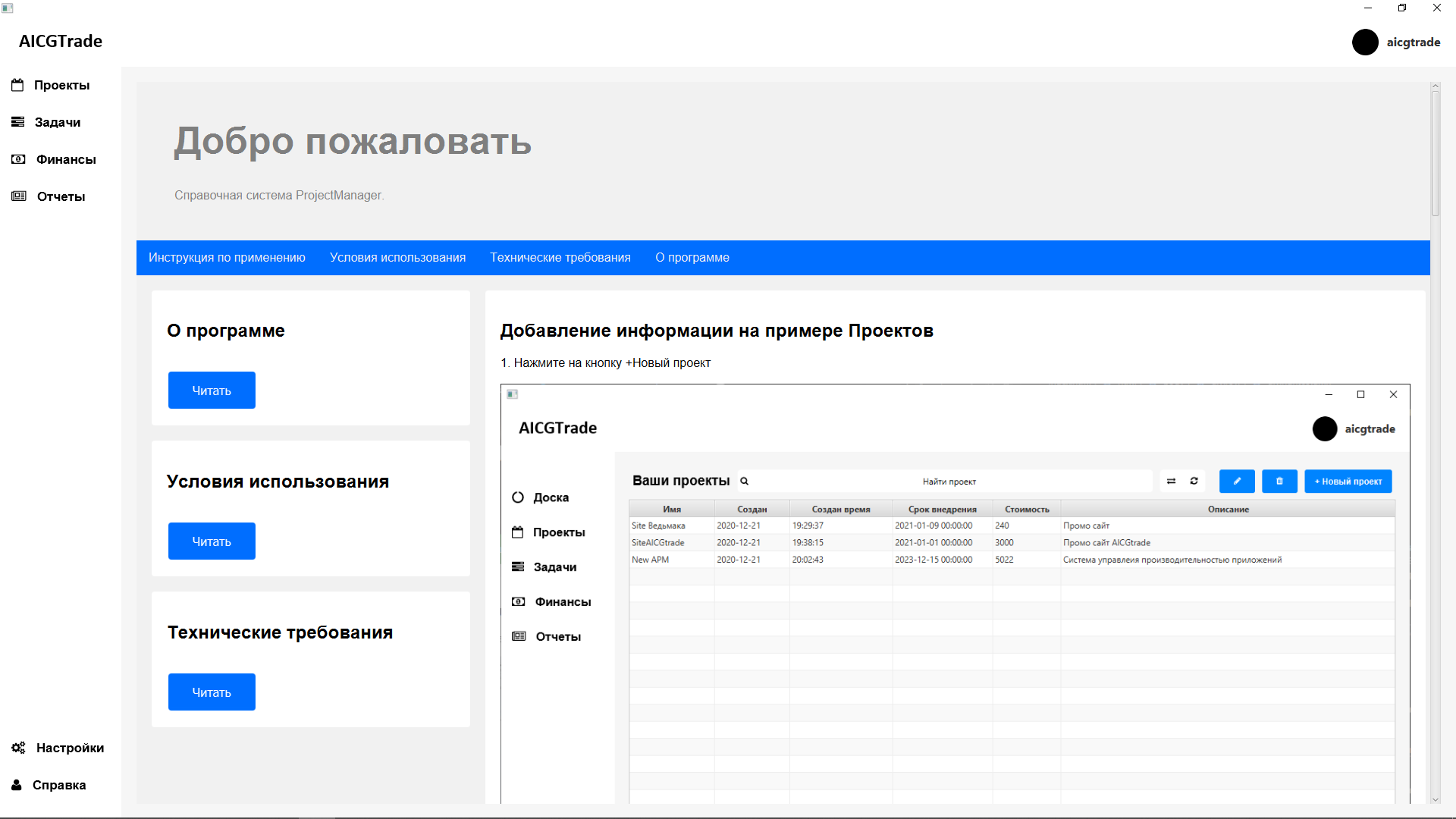


Рисунок 9.15 – Вкладка «Справка»

Таким образом, мы рассмотрели основной функционал приложения и протестировали его работу.

# Заключение

Прогресс не стоит на месте, поэтому всегда актуально создавать что-то новое, например, новые писать программы, которые упростят и ускорят работу человека и у сведут ручную работ к минимуму.

Автоматизация различных сфер жизни человека является очень значимой и важной, так как сокращает время обработки различной документации и освобождает сотрудников от рутинной работы. Автоматизация правления проектами компании требует использования специального программного продукта. Так как специфика деятельности в данной сфере требует хранение больших массивов информации, а также их систематизацию, необходимо создание баз данных и систем по их управлению, чтобы обрабатывать максимальное количество информации за минимальное время.

Подводя итоги, можно сказать, что созданная программа разработана для облегчения и удобства в работе с базами данных в отделах разработки и менеджмента предприятий и организациях. Приложение приводит в систему данные, позволяет обеспечить централизованное хранение информации, уменьшает затраты времени на ввод данных и их удаление.

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

# Список использованных источников

[1] Васильев А.Н. Java. – СПб «Питер», 2011. – 400 с.

[2] Блинов И.Н. «Java. Промышленное программирование. – Минск: «УниверсалПресс», 2007г. – 704с.

[3] Г. Буч, Д. Рамбо, А. Джекобсон. Язык UML Руководство пользователя.

[4] grandars.ru [Электронный ресурс]. – Электронные данные. – Режим доступа:http://www.grandars.ru/student/buhgalterskiy-uchet/uchet-osnovnyh-sredstv.html.

[5] ta-aspect.by [Электронный ресурс]. – Электронные данные. – Режим доступа:http://ta-aspect.by/Amortizaciya-osnovnyh-proizvodstvennyh-fondov.

[6] tic.tsu.ru [Электронный ресурс]. – Электронные данные. – Режим доступа:http://tic.tsu.ru/www/uploads/fma/2.4.2.htm.

[7] bourabai.kz [Электронный ресурс]. – Электронные данные. – Режим доступа:http://bourabai.kz/dbt/MySQL0109.htm.

# Приложение А

**(обязательное)**

**Модели представления системы (к разделу 6)**



Рисунок A.1 – Диаграмма состояний (Statechart diagram)

Продолжение приложения А



Рисунок A.2 – Диаграмма последовательностей (Sequence diagram)

Продолжение приложения А

Рисунок А.3.1 – Диаграмма классов

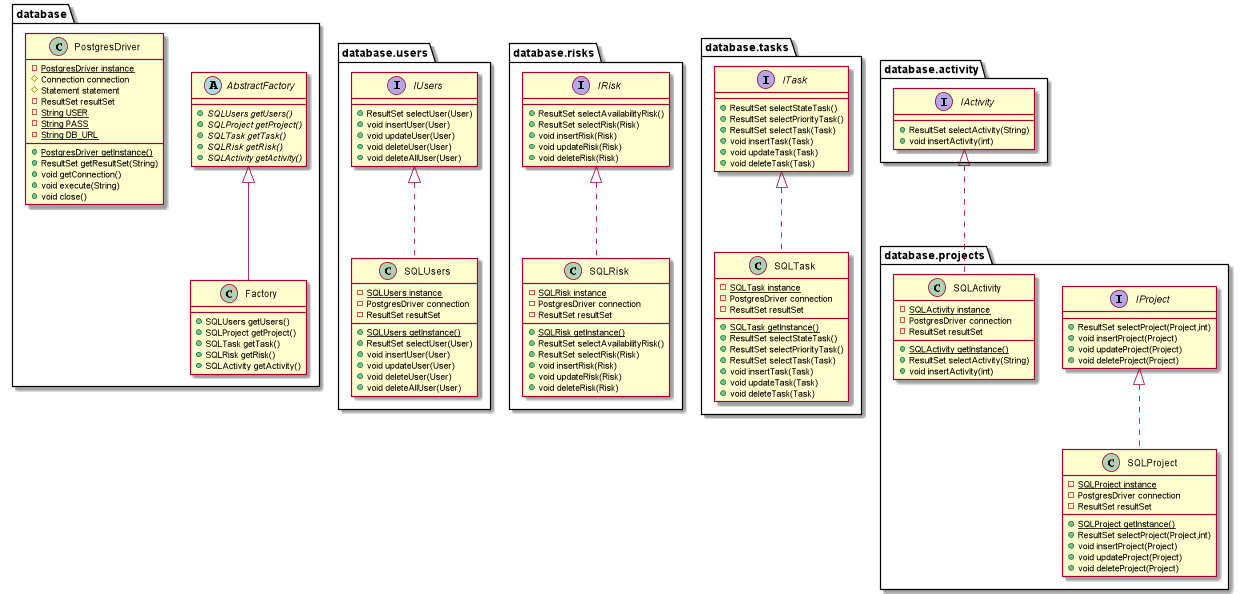
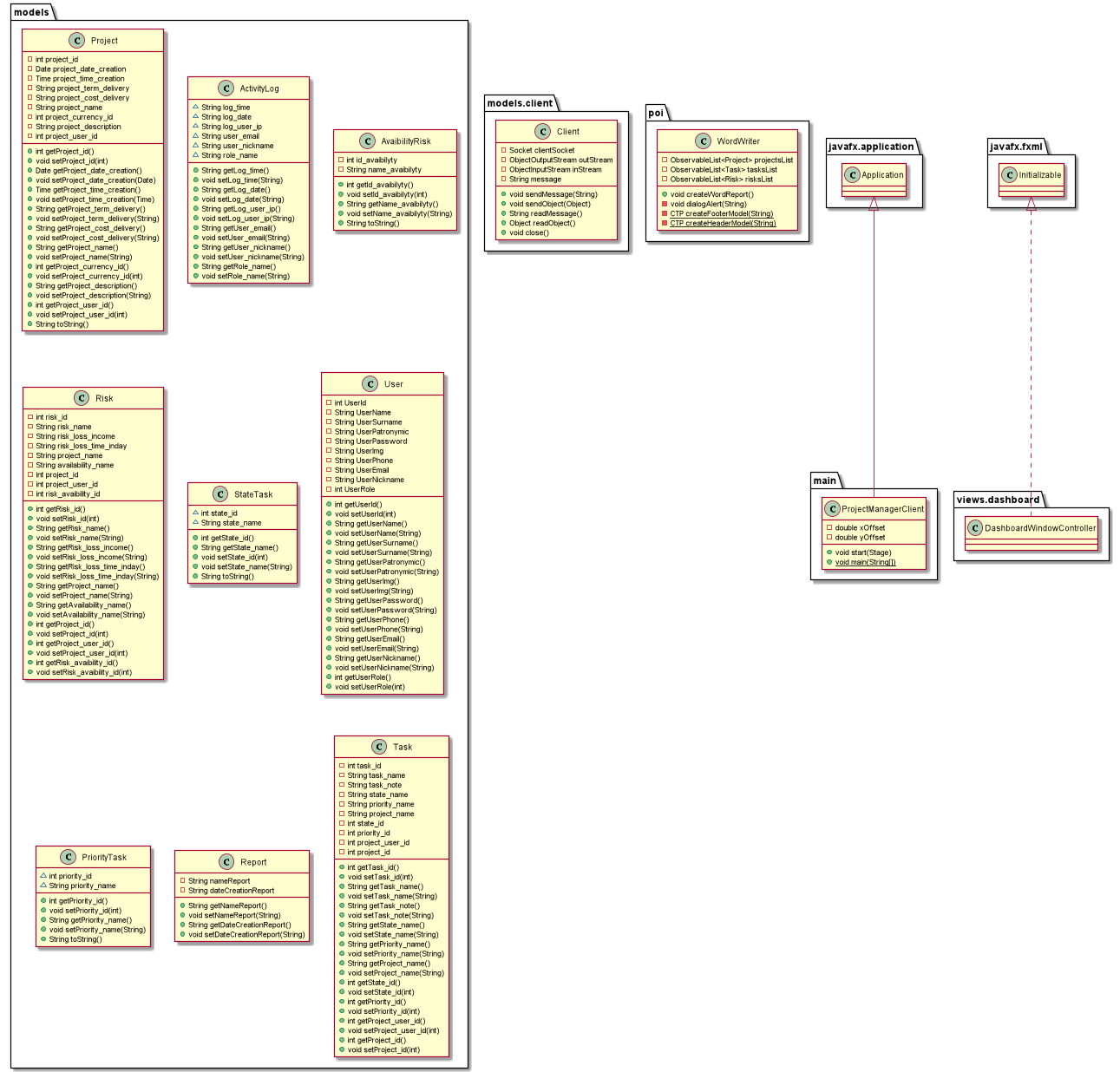


Рисунок А.3.2 – Диаграмма продолжение диаграммы классов



Продолжение приложения А



Рисунок A.4 – Диаграмма компонентов (Сomponent diagram)

# Приложение В

**(обязательное)**

**Скрипт создания БД**

DROP VIEW IF EXISTS projects\_view;

DROP VIEW IF EXISTS tasks\_view;

DROP VIEW IF EXISTS risks\_view;

DROP VIEW IF EXISTS activity\_log\_view;

DROP TABLE IF EXISTS activity,availability\_risk,currencies,priority\_tasks,projects,risks,roles,states\_tasks,tasks,users;

CREATE TABLE roles(

ROLE\_ID SERIAL PRIMARY KEY,

ROLE\_NAME CHARACTER VARYING( 15 ) UNIQUE NOT NULL

);

CREATE TABLE availability\_risk(

AVAILABILITY\_ID SERIAL PRIMARY KEY,

AVAILABILITY\_NAME CHARACTER VARYING( 4 ) UNIQUE NOT NULL

);

CREATE TABLE currencies(

CURRENCY\_ID SERIAL PRIMARY KEY,

CURRENCY\_EXCHANGE\_RATE NUMERIC,

CURRENCY\_NAME CHARACTER VARYING( 3 ) UNIQUE NOT NULL

);

CREATE TABLE states\_tasks(

STATE\_ID SERIAL PRIMARY KEY,

STATE\_NAME CHARACTER VARYING( 15 ) UNIQUE NOT NULL

);

CREATE TABLE priority\_tasks(

PRIORITY\_ID SERIAL PRIMARY KEY,

PRIOROTY\_NAME CHARACTER VARYING( 10 ) UNIQUE NOT NULL

);

CREATE TABLE users(

USER\_ID SERIAL PRIMARY KEY,

USER\_NAME TEXT,

USER\_SURNAME TEXT,

USER\_PATRONYMIC TEXT,

USER\_PASSWORD CHARACTER VARYING( 8 ) NOT NULL,

USER\_IMG TEXT,

USER\_PHONE CHARACTER VARYING( 100 ) UNIQUE NULL,

USER\_EMAIL CHARACTER VARYING( 255 ) UNIQUE NULL,

USER\_NICKNAME CHARACTER VARYING( 25 ) UNIQUE NOT NULL,

--FK-----------------------------------

USER\_ROLE\_ID INTEGER REFERENCES roles(ROLE\_ID) DEFAULT 3 NOT NULL

);

CREATE TABLE activity(

LOG\_ID SERIAL,

LOG\_TIME TIME,

LOG\_DATE DATE,

LOG\_USER\_IP CIDR NOT NULL,

--FK-----------------------------------

LOG\_USER\_ID INTEGER REFERENCES users(USER\_ID) NOT NULL

);

CREATE TABLE projects(

PROJECT\_ID SERIAL PRIMARY KEY,

PROJECT\_DATE\_CREATION DATE NOT NULL,

PROJECT\_TIME\_CREATION TIME NOT NULL,

PROJECT\_TERM\_DELIVERY DATE,

PROJECT\_COST\_DELIVERY NUMERIC,

PROJECT\_NAME CHARACTER VARYING( 255 ) NOT NULL,

PROJECT\_DESCRIPTION CHARACTER VARYING( 255 ),

--FK-----------------------------------

PROJECT\_CURRENCY\_ID INTEGER REFERENCES currencies(CURRENCY\_ID),

PROJECT\_USER\_ID INTEGER REFERENCES users(USER\_ID) NOT NULL

);

CREATE TABLE risks(

RISK\_ID SERIAL PRIMARY KEY,

RISK\_LOSS\_INCOME NUMERIC,

RISK\_LOSS\_TIME\_INDAY INTEGER,

RISK\_NAME TEXT,

--FK-----------------------------------

RISK\_AVAILABILITY\_ID INTEGER REFERENCES availability\_risk(AVAILABILITY\_ID) NOT NULL,

RISK\_PROJECT\_ID INTEGER REFERENCES projects(PROJECT\_ID) NOT NULL

);

CREATE TABLE tasks(

TASK\_ID SERIAL PRIMARY KEY,

TASK\_NAME CHARACTER VARYING( 255 ) NOT NULL,

TASK\_NOTE TEXT,

--FK-----------------------------------

TASK\_STATE\_ID INTEGER REFERENCES states\_tasks(STATE\_ID) NOT NULL,

TASK\_PRIORITY\_ID INTEGER REFERENCES priority\_tasks(PRIORITY\_ID) NOT NULL,

TASK\_PROJECT\_ID INTEGER REFERENCES projects(PROJECT\_ID) NOT NULL

);

INSERT INTO roles(ROLE\_NAME) VALUES('Администратор'),('Модератор'),('Пользователь');

INSERT INTO availability\_risk(AVAILABILITY\_NAME) VALUES('Есть'),('Нет');

INSERT INTO currencies(CURRENCY\_EXCHANGE\_RATE,CURRENCY\_NAME) VALUES(2.53,'USD'),(3.07,'EUR'),(3.44,'RUB'),(19.387,'BTC');

INSERT INTO states\_tasks(STATE\_NAME) VALUES('Инициирование'),('Планировние'),('Разработка'),('Одобрение'),('Закрыт');

INSERT INTO priority\_tasks(PRIOROTY\_NAME) VALUES('Высокий'),('Средний'),('Низкий');

CREATE VIEW projects\_view AS SELECT

projects.project\_id,

projects.project\_date\_creation,

projects.project\_time\_creation,

projects.project\_term\_delivery,

projects.project\_cost\_delivery,

projects.project\_name,

projects.project\_description,

currencies.currency\_exchange\_rate,

currencies.currency\_name,

projects.project\_user\_id

FROM projects

INNER JOIN currencies ON currencies.currency\_id = projects.project\_currency\_id ;

--RISKS VIEW

CREATE VIEW risks\_view AS SELECT

risk\_id,

risk\_name,

risk\_loss\_income,

risk\_loss\_time\_inday,

projects.project\_name,

availability\_risk.availability\_name,

availability\_risk.availability\_id,

projects.project\_id,

projects.project\_user\_id

FROM risks

INNER JOIN availability\_risk ON availability\_risk.availability\_id = risk\_availability\_id

INNER JOIN projects ON projects.project\_id = risk\_project\_id;

---TASKS VIEW

CREATE VIEW tasks\_view AS SELECT task\_id,

task\_name,

task\_note,

states\_tasks.state\_name,

priority\_tasks.prioroty\_name,

projects.project\_name,

states\_tasks.state\_id,

priority\_tasks.priority\_id,

projects.project\_user\_id,

projects.project\_id

FROM tasks

INNER JOIN states\_tasks ON states\_tasks.state\_id = task\_state\_id

INNER JOIN priority\_tasks ON priority\_tasks.priority\_id = task\_priority\_id

INNER JOIN projects ON projects.project\_id = task\_project\_id;

CREATE VIEW activity\_log\_view AS SELECT

log\_time,

log\_date,

log\_user\_ip,

users.user\_email,

users.user\_nickname,

roles.role\_name

FROM activity

INNER JOIN users ON users.user\_id = activity.log\_user\_id

INNER JOIN roles ON roles.role\_id = users.user\_role\_id;

# Приложение Г

**(обязательное)**

**Выборочный листинг кода**

**Файл ProjectManagerClient.java**

package main;  
  
import javafx.application.Application;  
import javafx.event.EventHandler;  
import javafx.scene.Scene;  
import javafx.scene.input.MouseEvent;  
import javafx.stage.Stage;  
import javafx.stage.StageStyle;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Parent;  
import javafx.scene.paint.Color;  
  
  
public class ProjectManagerClient extends Application {  
  
 private double xOffset = 0;  
 private double yOffset = 0;  
  
 @Override  
 public void start(Stage primaryStage){  
 Parent root;  
 try {  
  
 root = FXMLLoader.load(getClass().getResource("../views/authorization/LoginWindow.fxml"));  
 root.setOnMousePressed(new EventHandler<MouseEvent>() {  
  
 @Override  
 public void handle(MouseEvent event) {  
 xOffset = event.getSceneX();  
 yOffset = event.getSceneY();  
 }  
 });  
  
 root.setOnMouseDragged(new EventHandler<MouseEvent>() {  
 @Override  
 public void handle(MouseEvent event) {  
 primaryStage.setX(event.getScreenX() - xOffset);  
 primaryStage.setY(event.getScreenY() - yOffset);  
 }  
 });  
  
 Scene scene = new Scene(root);  
 primaryStage.setTitle("Hello World");  
 primaryStage.initStyle(StageStyle.UNDECORATED);  
 primaryStage.setScene(scene);  
 scene.setFill(Color.TRANSPARENT);  
 primaryStage.show();  
  
 }catch (Exception e){  
 e.getStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

**Файл PostgresDriver.java**

package database;  
  
import models.User;  
  
import java.sql.\*;  
import java.util.ArrayList;  
  
public class PostgresDriver {  
  
 private static PostgresDriver *instance*;  
 protected Connection connection;  
 protected Statement statement;  
 private ResultSet resultSet;  
  
 private static final String *USER* = "postgres";  
 private static final String *PASS* = "1";  
 private static final String *DB\_URL* = "jdbc:postgresql://127.0.0.1:5432/ProjectManager";  
  
 public PostgresDriver() {  
 getConnection();  
 }  
  
 public static synchronized PostgresDriver getInstance() {  
 if (*instance* == null) {  
 *instance* = new PostgresDriver();  
 }return *instance*;  
 }  
  
 public ResultSet getResultSet(String str) throws SQLException {  
 return this.resultSet = statement.executeQuery(str);  
 }  
  
 public void getConnection(){  
 System.*out*.println("Connection to PostgreSQL JDBC");  
  
 try {  
 Class.*forName*("org.postgresql.Driver");  
 } catch (ClassNotFoundException e) {  
 System.*out*.println("PostgreSQL JDBC Driver is not found. Include it in your library path ");  
 e.printStackTrace();  
 }  
  
 System.*out*.println("PostgreSQL JDBC Driver successfully connected");  
  
 try {  
 connection = DriverManager.*getConnection*(*DB\_URL*, *USER*, *PASS*);  
 statement = connection.createStatement();  
 } catch (SQLException e) {  
 System.*out*.println("Connection Failed");  
 e.printStackTrace();  
 }  
  
 if (connection != null) {  
 System.*out*.println("You successfully connected to database now");  
 } else {  
 System.*out*.println("Failed to make connection to database");  
 }  
  
 }  
  
 public void execute(String query) {  
 try {  
 statement.execute(query);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public void close() {  
 try {  
 connection.close();  
 } catch (SQLException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 }  
}