Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Рефакторинг и оптимизация программного кода

Отчет

по результатам выполнения лабораторных работ  
и заданий к практическим занятиям

|  |  |  |
| --- | --- | --- |
| Проверил |  | А.В. Шелест |
|  | (подпись) |  |
| зачтено |  |  |
|  | (дата защиты) |  |
|  |  |  |
| Выполнил |  | В.С. Голуб  гр. 214371 |
|  | (подпись) |  |

**СОДЕРЖАНИЕ**

[1 АРХИТЕКТУРА ПРОГРАММНОГО СРЕДСТВА 4](#_Toc195898160)

[2 ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО 12](#_Toc195898161)

[ИНТЕРФЕЙСА ПС 12](#_Toc195898162)

[3 РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ ПС 14](#_Toc195898163)

[4 СПРОЕКТИРОВАТЬ СХЕМУ БД И ПРЕДСТАВИТЬ 15](#_Toc195898164)

[ОПИСАНИЕ ЕЕ СУЩНОСТЕЙ И ИХ АТРИБУТОВ 15](#_Toc195898165)

[5 ДЕТАЛИ РЕАЛИЗАЦИИ ПС ЧЕРЕЗ UML-ДИАГРАММЫ 19](#_Toc195898166)

[6 ДОКУМЕНТАЦИЯ К ПС С OPEN API 24](#_Toc195898167)

[7 РЕАЛИЗАЦИЯ СИСТЕМЫ АУТЕНТИФИКАЦИИ И 26](#_Toc195898168)

[АВТОРИЗАЦИИ ПОЛЬЗОВАТЕЛЕЙ ПС И 26](#_Toc195898169)

[МЕХАНИЗМОВ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ 26](#_Toc195898170)

[ДАННЫХ 26](#_Toc195898171)

[8 UNIT- И ИНТЕГРАЦИОННЫЕ ТЕСТЫ 27](#_Toc195898172)

[9 ОПИСАНИЕ ПРОЦЕССА РАЗВЕРТЫВАНИЯ ПС 31](#_Toc195898173)

[10 РАЗРАБОТКА РУКОВОДСТВА ПОЛЬЗОВАТЕЛЯ 33](#_Toc195898174)

[ВЫВОД 39](#_Toc195898175)

**ССЫЛКИ НА РЕПОЗИТОРИИ**

[ProgressiveKid/GolubV\_214371\_RIOPK\_Server](https://github.com/ProgressiveKid/GolubV_214371_RIOPK_Server)

[ProgressiveKid/GolubV\_214371\_RIOPK\_Client](https://github.com/ProgressiveKid/GolubV_214371_RIOPK_Client)

[ProgressiveKid/GolubV\_214371\_RIOPK\_Spec](https://github.com/ProgressiveKid/GolubV_214371_RIOPK_Spec)

# **1 АРХИТЕКТУРА ПРОГРАММНОГО СРЕДСТВА**

Диаграмма вариантов использования для программного средства представлена на рисунке 1.

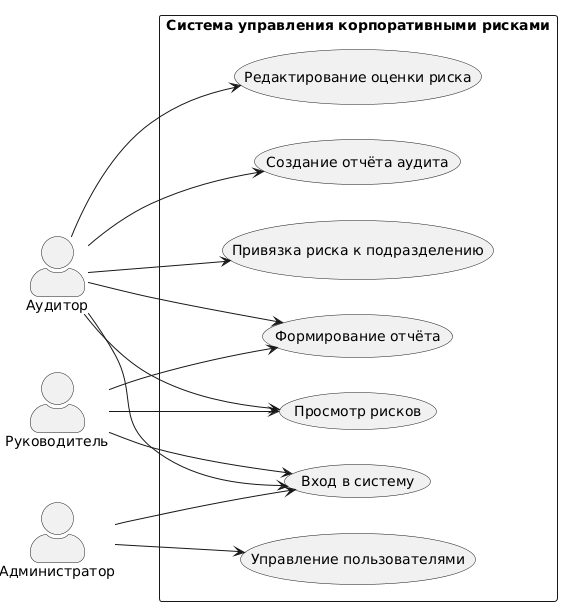


Рисунок 1.1 – Диаграмма вариантов использования

Диаграмма вариантов использования — диаграмма, отражающая отно-шения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмма вариантов использования описывает взаимодействие актеров с системой и ключевые сценарии использования. В нашу систему входят следующие основные варианты использования:

1. Просмотр рисков: пользователи могут получить актуальную информацию о корпоративных рисках в системе.
2. Оценка и редактирование рисков: пользователи могут создавать и редактировать оценки рисков, в том числе привязывать риски к соответствующим подразделениям.
3. Генерация отчетов: пользователи могут формировать отчеты о рисках для анализа и принятия управленческих решений.
4. Управление пользователями: администратор может создавать, редактировать и удалять учетные записи пользователей, а также назначать роли и уровни доступа.
5. Авторизация и регистрация пользователей: система позволяет пользователю зарегистрироваться, войти в систему и получить доступ к функционалу в зависимости от его роли.

Роли:

Аудитор: выполняет оценку рисков, привязывает их к подразделениям и генерирует отчеты. Может просматривать информацию о рисках и аудиторских проверках.

Руководитель: просматривает риски и генерирует отчеты для принятия решений. Имеет доступ к общей информации.

Администратор: управляет учетными записями пользователей, назначает роли, осуществляет контроль доступа к системе. Также может выполнять вход в систему для выполнения административных задач.

Процесс регистрации доступен для новых пользователей, что позволяет им создать учетную запись и начать работать с системой. Авторизация пользователей контролирует доступ к различным частям системы в зависимости от их роли.

C4-модель архитектуры

Диаграмма контекста отображает, как Система управления корпоративными рисками взаимодействует с внешними системами и пользователями (рис 1.2)

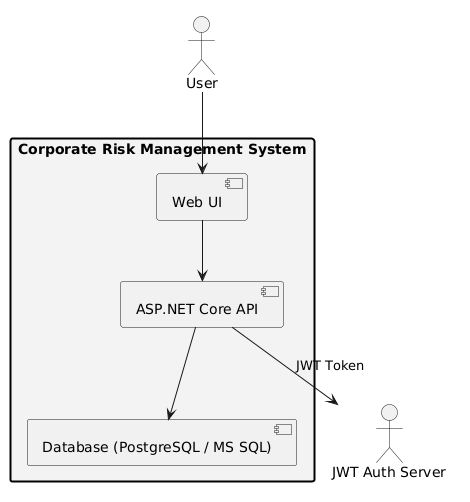


Рисунок 1.2 – Диаграмма контекста

Контейнерная диаграмма показывает внутреннюю структуру системы — её компоненты, как они взаимодействуют между собой, и с внешними системами.

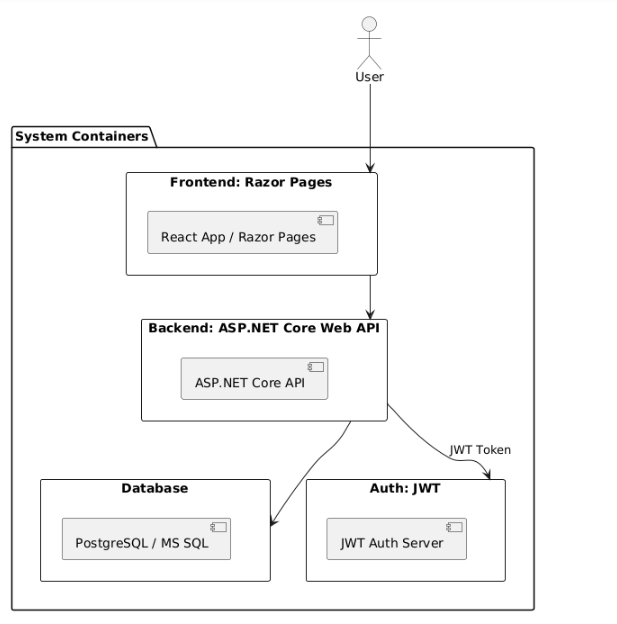


Рисунок 1.3 – Диаграмма контекста

Диаграмма компонентов детализирует ключевые компоненты системы, такие как контроллеры, сервисы и репозитории, их связи и взаимодействия.

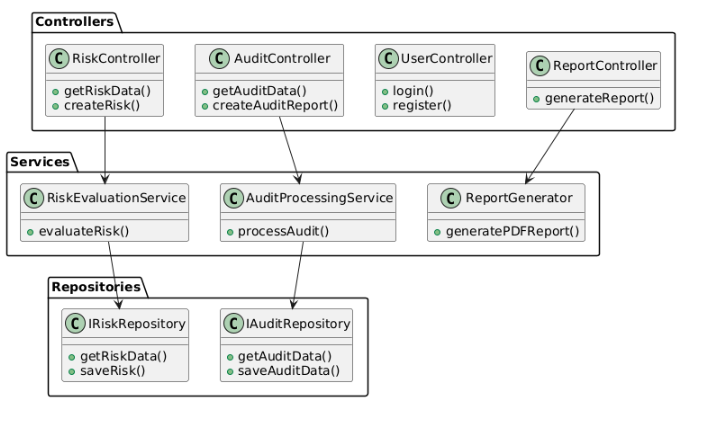


Рисунок 1.4 – Диаграмма контекста

Система дизайна – набор элементов и их стилей оформления. Цель системы дизайна пользовательского интерфейса состоит в том, чтобы обеспечить эффективное взаимодействие пользователей с программным средством, удовлетворяя их потребности и ожидания. Эта система охватывает следующие аспекты:

1. Макеты и компоненты: определяет структуру интерфейса, включая расположение элементов, размеры и пропорции, а также использование различных компонентов, таких как кнопки, поля ввода и т. д.

2. Интерактивность: обеспечивает пользователей возможностью взаимодействия с интерфейсом через жесты, клики и другие действия, а также реагирует на эти действия соответствующим образом, обеспечивая обратную связь и понимание того, что происходит.

3. Адаптивность: учитывает различные устройства и разрешения экранов, чтобы обеспечить оптимальный пользовательский опыт на различных платформах, включая компьютеры, планшеты и мобильные устройства.

Планы расположения элементов на странице для различных устройств помогают обеспечить оптимальное отображение и удобство использования вашего веб-сайта или приложения на различных устройствах, таких как компьютеры, планшеты и мобильные телефоны. Эти планы обычно включают в себя адаптивный дизайн и медиа-запросы, которые позволяют элементам интерфейса переходить между разными компоновками в зависимости от размера экрана устройства. Кроме того, они учитывают уникальные особенности и ограничения каждого устройства, такие как разрешение экрана, размер и расположение элементов управления. Это обеспечивает единое и приятное пользовательское взаимодействие независимо от устройства, которое использует пользователь

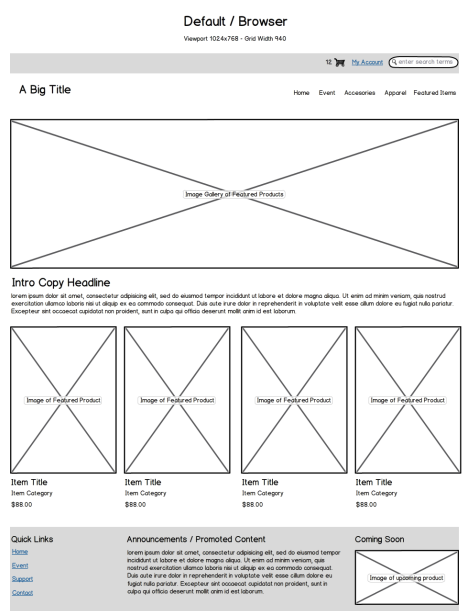


Рисунок 1.5 – План размещения элементов интерфейса в браузере

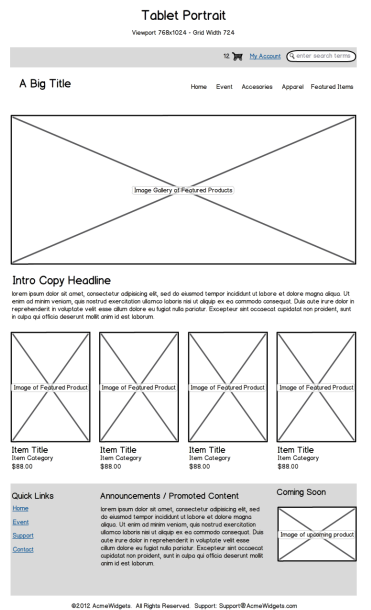


Рисунок 1.6 – План размещения элементов интерефейса на планшете

В качестве технологии для написания сервера было принято решение использовать фреймворк *ASP NET Core Web API*. База данных используется *MSSQLr*. Для клиентского языка программирования выбран Razor Page. Этот выбор обусловлен высокой производительностью, обширным экосистемом инструментов и библиотек для разработки веб-приложений, а также широкой популярностью и поддержкой сообщества разработчиков

Выбор ASP.NET Core обоснован высокой производительностью, широкими возможностями интеграции с экосистемой .NET и инструментами Microsoft, а также доступностью обширной документации и поддержки сообщества разработчиков. ASP.NET Core предоставляет надежный и эффективный инструментарий для разработки веб-приложений, а также обладает возможностью развертывания на различных платформах, что делает его привлекательным выбором для широкого спектра проектов.

Структура архитектуры проекта представлена на рисунке 1.11.

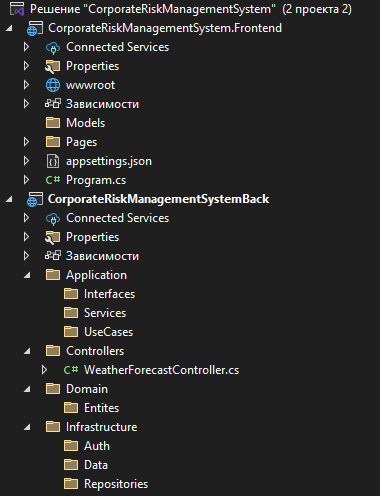


Рисунок 1.7 – Структура проекта

Проект построен по принципам Clean Architecture, что позволяет чётко разделить логику и ответственность каждого слоя, обеспечивая удобство сопровождения, тестируемость и масштабируемость. Структура проекта делится на две части: Backend (серверная часть) и Frontend (клиентская часть).

Backend (Серверная часть)

Серверная часть проекта организована с учётом принципов Clean Architecture, и включает в себя несколько слоёв, каждый из которых отвечает за свою область. Вся бизнес-логика и взаимодействие с данными происходят в серверной части.

API: Папка для контроллеров, которые обрабатывают HTTP-запросы и вызывают соответствующую логику из слоёв Application и Domain. Пример контроллеров: RiskController, AuditController — они отвечают за операции, связанные с рисками и аудитом.

Application: Этот слой отвечает за бизнес-логику приложения и представляет собой интерфейсы и реализацию конкретных случаев использования (Use Cases). В папке Interfaces находятся интерфейсы для сервисов и репозиториев, которые определяют контракты для работы с данными и бизнес-операциями. В папке UseCases реализованы конкретные случаи использования, такие как создание нового риска или генерация отчёта. В папке Services находится бизнес-логика, которая реализует функциональность приложения.

Domain: В слое Domain находятся сущности (например, Risk, AuditReport, Department), которые представляют собой основные объекты, с которыми работает бизнес-логика. Если требуется, здесь могут быть реализованы значимые объекты (Value Objects), например, RiskLevel или AuditStatus.

Infrastructure: Этот слой содержит реализации взаимодействия с внешними системами, такими как база данных или сервисы аутентификации. В папке Data реализован доступ к данным через Entity Framework Core, а также миграции для работы с базой данных. В папке Repositories находятся репозитории, которые обеспечивают доступ к данным для каждой из сущностей. Папка Auth отвечает за реализацию аутентификации и авторизации пользователей, используя, например, JWT.

Program.cs: Это точка входа в серверное приложение, где настраиваются все необходимые зависимости, такие как контейнеры DI (dependency injection), маршрутизация и обработка запросов.

Frontend (Клиентская часть)

Клиентская часть проекта представляет собой веб-приложение, использующее Razor Pages для создания интерфейса. Это упрощает взаимодействие с пользователем, обеспечивая чёткое разделение логики и интерфейса.

Pages: Здесь находятся Razor Pages, которые отвечают за отображение данных и взаимодействие с пользователем. Страницы могут быть такими, как: Index.cshtml — главная страница. Login.cshtml — страница входа для пользователей. RiskList.cshtml — страница для просмотра списка рисков. Также здесь находится \_ViewImports.cshtml, который применяется для общей настройки представлений, таких как подключение библиотек и общих стилей.

Models: В этой папке находятся модели данных, которые используются для отображения информации на страницах. Например, это могут быть модели для рисков (RiskModel.cs) или пользователей (UserModel.cs), которые передаются между контроллерами и представлениями.

wwwroot: Папка для статических файлов: CSS, JS, изображения. Эти файлы обеспечивают стиль и функциональность на клиентской части. Они обрабатываются непосредственно браузером.

Program.cs: Точка входа в приложение на стороне клиента. Здесь настраивается всё, что касается работы с Razor Pages.

Общий подход к архитектуре

Проект разделён на два репозитория: один для Backend, другой для Frontend, что позволяет чётко разграничить клиентскую и серверную части.

Clean Architecture в проекте:

Каждый слой в архитектуре имеет свою четкую зону ответственности. Backend слой разделён на несколько слоёв, начиная от API контроллеров, которые принимают запросы, до слоя инфраструктуры, который взаимодействует с внешними сервисами. Frontend слой сосредоточен на создании интерфейса для пользователей, взаимодействующего с сервером через Razor Pages.

Преимущества такой архитектуры:

Тестируемость: Каждый слой приложения может быть протестирован независимо от других. Логика бизнес-операций тестируется в слое Application, а доступ к данным — в слое Infrastructure.

Масштабируемость: Архитектура позволяет масштабировать приложение, разделяя его на независимые компоненты. Это полезно, например, при добавлении новых сервисов или изменении базовых сущностей.

Читаемость и поддерживаемость: Благодаря разделению на слои проект остаётся удобным для сопровождения, особенно когда он растёт и требуется добавление новых функциональностей.

**2 ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО**

**ИНТЕРФЕЙСА ПС**

User Flow диаграмма — это визуальное представление пути, который проходят пользователи в приложении, начиная с момента входа и заканчивая выполнением конкретных действий.

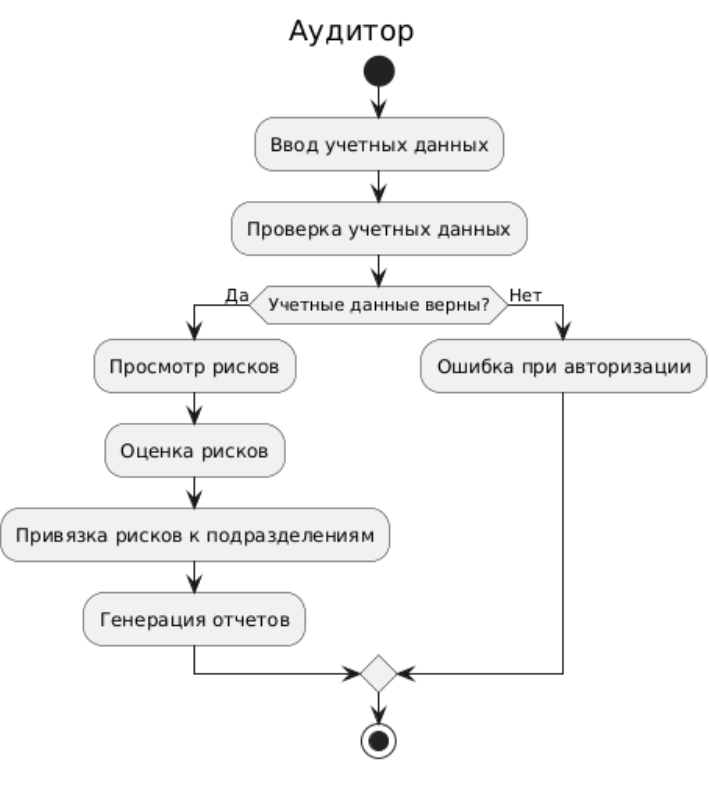


Рисунок 2.1 – user flow аудитора

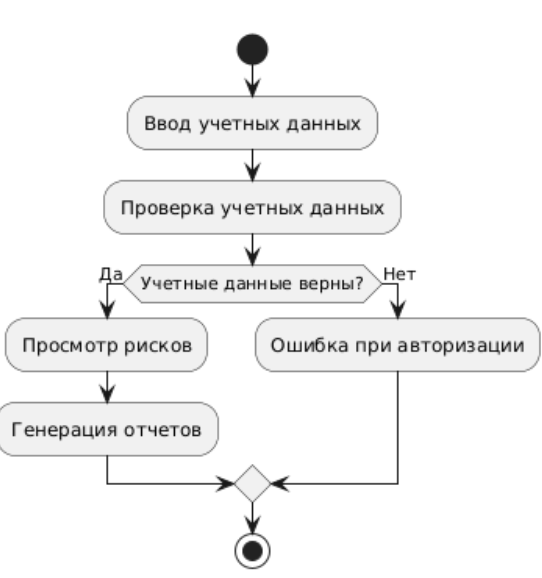


Рисунок 2.2 – user flow руководителя

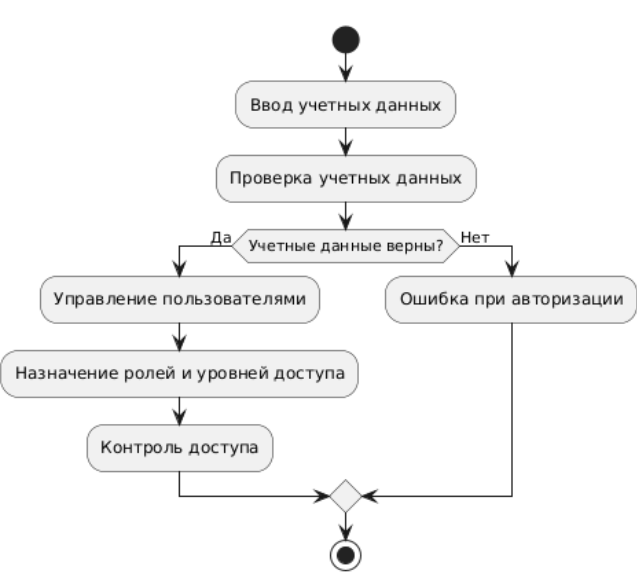


Рисунок 2.3 – user flow администратора

**3 РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ ПС**

Для клиентского языка программирования выбран Razor Page. Этот выбор обусловлен высокой производительностью, обширным экосистемом инструментов и библиотек для разработки веб-приложений, а также широкой популярностью и поддержкой сообщества разработчиков. JavaScript с Razor Page предоставляет эффективные инструменты для создания интерактивных пользовательских интерфейсов и обеспечения отличного пользовательского опыта.

Структура архитектуры клиентской части представлена на рисунке 3.

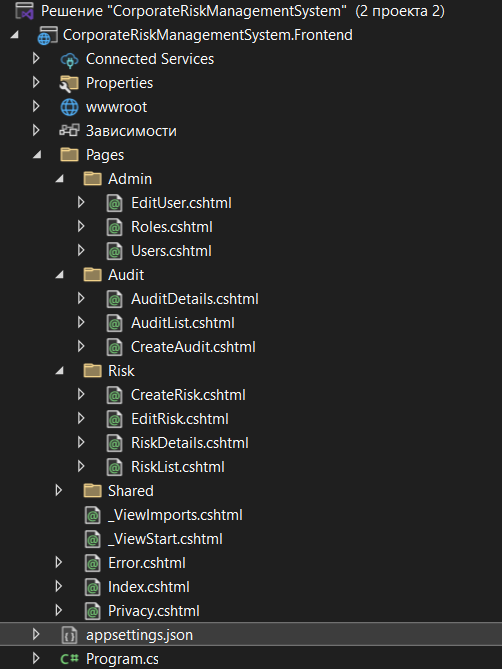


Рисунок 3.1 – Схема клиентской части проекта

**4 СПРОЕКТИРОВАТЬ СХЕМУ БД И ПРЕДСТАВИТЬ**

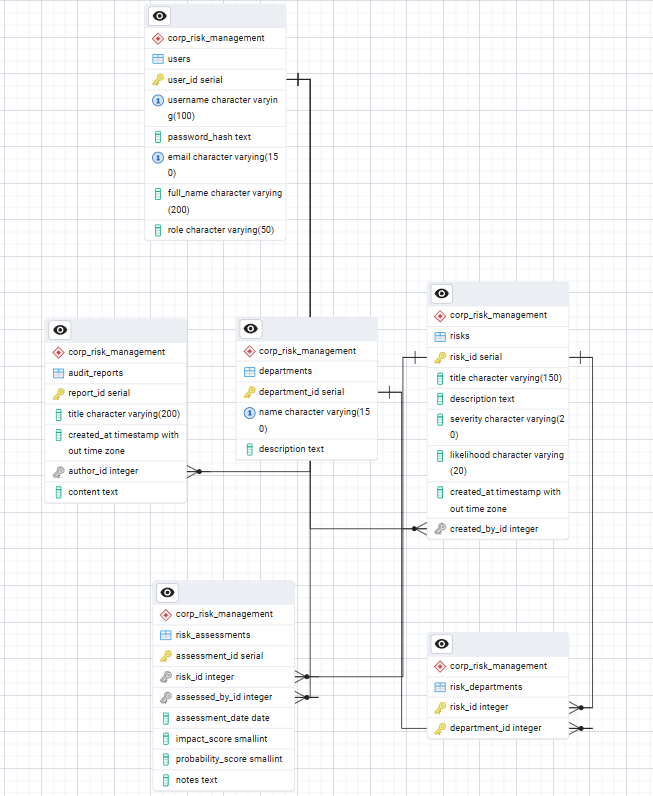
**ОПИСАНИЕ ЕЕ СУЩНОСТЕЙ И ИХ АТРИБУТОВ**

Предметной областью данного проекта является система управления корпоративными рисками на основе данных внутреннего аудита. Для формализации предметной области необходимо выделить ключевые бизнес-процессы, основные сущности и варианты использования системы различными категориями пользователей.

На основании анализа специфики данной области можно выделить следующие базовые сущности реализуемого программного средства:

1. Пользователь — представляет собой субъект, взаимодействующий с системой. Имеет роль (аудитор, руководитель, администратор), от которой зависит доступ к функционалу.
2. Риск — потенциальная угроза или событие, способное оказать влияние на деятельность организации. Может быть оценён, классифицирован и привязан к определённому подразделению.
3. Подразделение — структурная единица организации, к которой могут быть привязаны определённые риски или отчёты.
4. Отчёт аудита — документ, созданный в результате аудиторской проверки, содержащий выводы и оценки по выявленным рискам.
5. Оценка риска — результат анализа риска с указанием уровня воздействия и вероятности, устанавливаемый аудитором.

Эта формализация обеспечивает основу для проектирования архитектуры программной системы, её интерфейсов и схемы базы данных.

  
Рисунок 4 – Схема базы данных

Физическое описание модели данных охватывает информацию о пользователях, корпоративных рисках, подразделениях и отчетах, что позволяет эффективно управлять критичными для организации данными внутреннего аудита. Модель включает в себя как основные сведения о каждой сущности (например, имя пользователя, описание риска, дата отчета), так и взаимосвязи между ними, обеспечивающие целостность и достоверность информации при анализе и принятии управленческих решений.

Физическая модель данных соответствует требованиям третьей нормальной формы (3НФ) по следующим причинам:

1 Атомарные атрибаты: все поля таблиц приведены к атомарному виду — т.е. они не могут быть логически разделены без потери смысла. Пример: поля Имя, Фамилия, Email в сущности Пользователь не делятся далее.

2 Уникальные идентификаторы: каждая таблица содержит первичный ключ (Id), гарантирующий уникальность записей и целостность ссылочной логики.

3 Зависимость только от ключей: атрибуты зависят исключительно от первичных ключей, отсутствуют транзитивные зависимости. Пример: RiskId в таблице AuditReport указывает на конкретный риск, не содержит повторяющейся информации, зависящей от неключевых полей

Таблица 1 – Пользователи (User)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Id | Имя | Фамилия | Отчество | Email | Пароль (хэш) | Телефон | Роль |
| 1 | Влад | Голуб | Сергеевич | vlad.golub@example.com | $2a$10$... | +375291234567 | Аудитор |
| 2 | Елена | Иванова | Николаевна | elena.ivanova@example.com | $2a$10$... | +375331112233 | Администратор |

Таблица 2 – Подразделения (Department)

|  |  |  |
| --- | --- | --- |
| Id | Название | РуководительId (FK -> User.Id) |
| 1 | Финансовый отдел | 2 |
| 2 | Отдел внутреннего контроля | 1 |

Таблица 3 — Риски (Risk)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Название | Описание | Уровень риска | ПодразделениеId (FK -> Department.Id) |
| 1 | Потеря данных | Возможность утраты клиентской информации | Высокий | 2 |
| 2 | Ошибки в отчетности | Некорректное заполнение финансовых форм | Средний | 1 |

Таблица 4 — Отчёты аудита (AuditReport)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Id | Название | Дата проведения | RiskId (FK) | АудиторId (FK -> User.Id) | Комментарий |
| 1 | Аудит финансов | 2025-03-25 | 2 | 1 | Обнаружены несоответствия в расчетах |
| 2 | Аудит безопасности | 2025-04-02 | 1 | 1 | Необходимо улучшить систему резервного копирования |

Таблица 5 — Оценка риска (RiskAssessment)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Id | Уровень воздействия | Вероятность | Оценка (автомат.) | Дата | RiskId (FK) | Оценил (FK -> User.Id) |
| 1 | Высокий | Низкая | Средняя | 2025-03-24 | 1 | 1 |
| 2 | Средний | Средняя | Средняя | 2025-03-28 | 2 | 1 |

Схема базы данных соответствует третьей нормальной форме, так как все таблицы имеют уникальные идентификаторы, все не ключевые атрибуты зависят только от первичных ключей, и отсутствуют транзитивные зависимости. Это обеспечивает минимизацию избыточности данных и упрощает управление данными.

**5 ДЕТАЛИ РЕАЛИЗАЦИИ ПС ЧЕРЕЗ UML-ДИАГРАММЫ**

Для разработки программного средства управления корпоративными рисками и аудиторскими отчетами была выбрана клиент-серверная архитектура, с разделением на несколько слоев: API, бизнес-логика, доменная модель и инфраструктура. В данном подходе каждый слой имеет четко определенные задачи, что способствует модульности, расширяемости и поддерживаемости кода.

API слой: В этом слое размещены контроллеры API, такие как RiskController, AuditController, которые обрабатывают HTTP-запросы и взаимодействуют с соответствующими сервисами и бизнес-логикой. Контроллеры принимают запросы от клиента и передают данные в слой бизнес-логики, который реализует все необходимые операции. Контроллеры также обеспечивают правильный формат ответа и статус-коды.

Слой Application: Этот слой отвечает за бизнес-логику приложения. В нем размещаются:

Интерфейсы сервисов и репозиториев в папке Interfaces, которые описывают контракты для взаимодействия с данными и выполнения бизнес-операций.

UseCases — реализация конкретных случаев использования, например, CreateRiskUseCase, который инкапсулирует логику создания нового риска. Это помогает изолировать конкретные сценарии от других частей приложения.

Services — сервисы, реализующие бизнес-логику. Например, RiskService может быть ответственен за создание, обновление и удаление рисков, а также за обработку аудиторских отчетов.

Слой Domain: В доменном слое располагаются основные сущности и значимые объекты:

Entities — здесь находятся основные бизнес-объекты, такие как Risk, AuditReport, User, которые являются частью предметной области. Эти объекты отвечают за хранение и управление данными, связанными с рисками, аудиторскими отчетами и пользователями.

ValueObjects — для более сложных типов данных, которые не имеют собственного идентификатора, но важны для бизнес-логики, например, временные интервалы или сложные адресные структуры.

Слой Infrastructure: В этом слое реализованы все внешние взаимодействия с системой:

Data — работа с данными с использованием ORM, например, Entity Framework Core, для взаимодействия с реляционной базой данных. В этом слое обеспечивается работа с репозиториями, которые отвечают за доступ к данным.

Repositories — здесь находятся репозитории для работы с сущностями, например, RiskRepository, которые обеспечивают доступ к данным и взаимодействие с базой данных через запросы и операции с данными.

Auth — реализация аутентификации и авторизации, например, через JWT (JSON Web Token), для безопасного доступа к API. В этом слое реализуется проверка токенов и управление доступом на основе ролей пользователей.

В этой архитектуре используется паттерн Dependency Injection (DI) для обеспечения слабой связанности между слоями. Компоненты получают зависимости через конструктор, что упрощает тестирование и улучшает масштабируемость системы. Все зависимости и конфигурация сервисов регистрируются в Startup.cs или Program.cs.

Также в этом проекте активно используется паттерн Unit of Work через DbContext в Entity Framework Core. Этот паттерн гарантирует, что все операции с данными выполняются атомарно, что важно для поддержания целостности данных при обработке аудиторских отчетов и управлении рисками. Все изменения в базе данных обрабатываются транзакционно с помощью метода SaveChanges().

Сочетание этих паттернов, а также четкая структура слоев, обеспечивают гибкость, расширяемость и масштабируемость приложения. Архитектура ориентирована на создание системы, способной эффективно управлять рисками, аудитом, пользователями и отчетами, что особенно важно для обеспечения безопасности и прозрачности в организации.

Диаграмма последовательности на примере создания заказа на бронирование номера в системе покажет, как пользователь и управляющие команды передаются между объектами в процессе (рис 5.1).

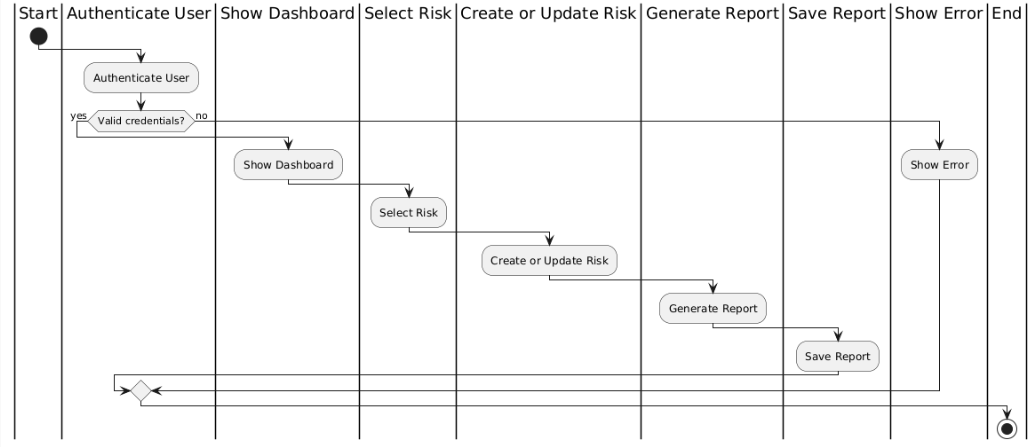


Рисунок. 5.1 – Диаграмма деятельности

Диаграмма деятельности используется для моделирования бизнес-процессов и потоков работ в системе. В отличие от диаграмм последовательности, которые показывают взаимодействие между объектами, диаграмма деятельности фокусируется на последовательности действий, выполняемых в рамках бизнес-процесса.

1. Диаграмма деятельности может быть использована для описания процесса создания и обработки рисков:
2. Начало процесса: Аудитор или другой пользователь инициирует создание нового риска.
3. Запрос данных: Вводится информация о риске, включая категорию, вероятности и описание.
4. Проверка данных: Программа проверяет правильность введенных данных (например, корректность данных о департаменте, связанного с риском).
5. Создание риска: Если данные верны, риск сохраняется в системе, иначе пользователю отображается ошибка.
6. Генерация отчета: После создания риска система генерирует отчет на основе введенных данных.
7. Завершение процесса: Риск сохраняется, отчет доступен для просмотра.

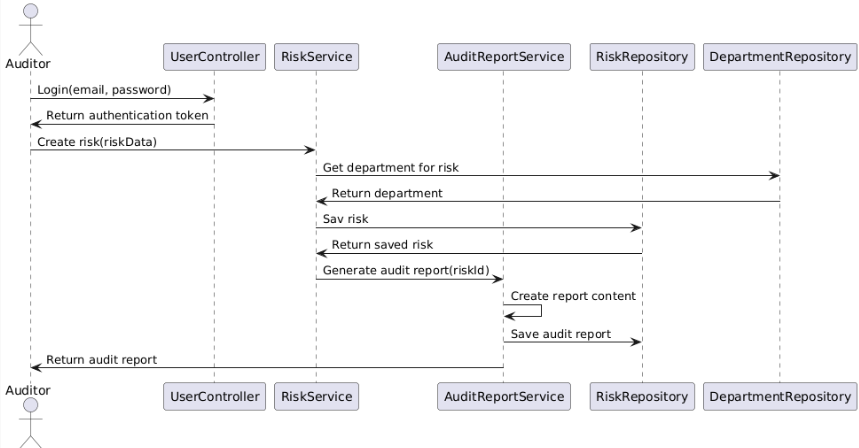


Рисунок. 5.2 – Диаграмма последовательности

Диаграмма последовательности описывает порядок взаимодействий между участниками системы для реализации определенной функции:

1. Аудитор выполняет вход в систему, получает токен.
2. Аудитор создает новый риск через Сервис рисков.
3. Сервис рисков взаимодействует с репозиториями для получения департамента и сохранения риска.
4. Сервис отчетов генерирует отчет по риску и сохраняет его в систему.
5. Диаграмма последовательности показывает, как данные и действия проходят через систему в процессе выполнения бизнес-операций.

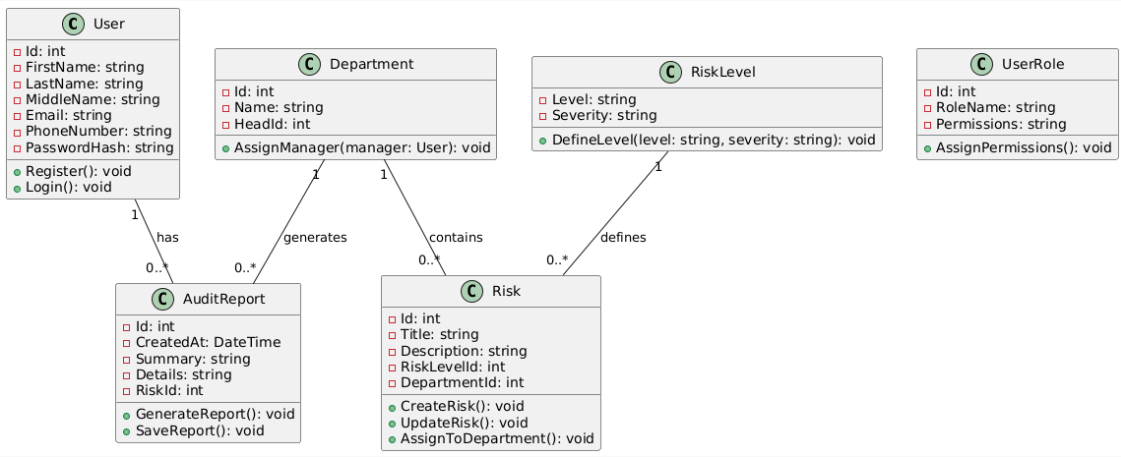


Рисунок. 5.3 – Диаграмма классов

Диаграмма классов описывает основные сущности системы и их атрибуты, а также отношения между этими сущностями (классы, их свойства и методы). В примере с системой управления рисками это:

1. Пользователь: содержит информацию о пользователе (имя, фамилия, электронная почта).
2. Риск: описание риска, его категории, вероятности и влияния.
3. Отчет: создается на основе данных о риске и включает описание и выводы.
4. Департамент: подразделение, к которому привязан риск.
5. Заказ: информация о бронировании.

# **6 ДОКУМЕНТАЦИЯ К ПС С OPEN API**

Разработанная документация предоставляет разработчикам полное описание всех методов API, необходимых для работы системы управления корпоративными рисками. Функционал охватывает процессы создания рисков, генерации аудиторских отчётов, проверки подразделений, управления пользователями и истории операций. Интерактивная версия спецификации доступна через Swagger UI и содержит примеры всех входных и выходных данных.

Ключевые разделы документации включают:

1. создание и редактирование рисков;
2. формирование отчётов по результатам аудита;
3. управление учетными записями и ролями пользователей;
4. просмотр истории зарегистрированных инцидентов и действий;

Пример описания метода API для создания риска:

POST /api/risks

summary: Создание нового риска

description: Добавляет риск в систему, связывая его с подразделением и аудиторским отчётом.

requestBody:

required: true

content:

application/json:

schema:

type: object

properties:

title:

type: string

description: Краткое название риска

example: "Недостаточный контроль доступа"

description:

type: string

example: "Несанкционированный доступ к данным отдела ИТ"

departmentId:

type: integer

example: 5

responses:

201:

description: Риск успешно создан

content:

application/json:

schema:

type: object

properties:

id:

type: integer

example: 42

status:

type: string

example: "created"

400:

description: Ошибка валидации

500:

description: Внутренняя ошибка сервера

Код соответствует следующим критериям:

Структурная чёткость: Контроллеры (например, RiskController) отвечают за маршрутизацию и валидацию, бизнес-логика реализуется в сервисах (RiskService, Ключевые архитектурные принципы: Структурная чёткость: контроллеры, такие как RiskController, отвечают за маршрутизацию и валидацию, бизнес-логика реализуется в сервисах, например, в RiskService и AuditService, а доступ к данным инкапсулирован в репозиториях, таких как RiskRepository и DepartmentRepository. Безопасность обеспечивается через JWT-аутентификацию, разграничение доступа по ролям (например, Auditor и Admin), а также через валидацию всех входных данных. Надёжность системы достигается благодаря централизованной обработке ошибок, логированию операций и сбоев, а также устойчивости к ошибкам внешних зависимостей, таких как отказ базы данных или стороннего сервиса. Тестируемость системы обеспечивается через модульное покрытие логики, включая юнит-тесты для таких сервисов, как RiskService и AuditService, а также интеграционные тесты для API с использованием xUnit

**7 РЕАЛИЗАЦИЯ СИСТЕМЫ АУТЕНТИФИКАЦИИ И**

**АВТОРИЗАЦИИ ПОЛЬЗОВАТЕЛЕЙ ПС И**

**МЕХАНИЗМОВ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ**

**ДАННЫХ**

Для реализации аутентификации и авторизации пользователей в системе используется механизм JSON Web Tokens (JWT). Данный подход позволяет безопасно управлять доступом к ресурсам приложения.

**ASP.NET Core Identity** используется для управления пользователями и ролями в системе. **JWT** применяется для создания и верификации токенов, позволяющих аутентифицировать пользователей.

**Конфигурация Identity**. В приложении настраивается контекст базы данных и служба Identity, что обеспечивает управление пользователями и ролями. Настройка аутентификации. Настраиваются параметры для обработки JWT токенов, такие как ключи подписи, валидаторы и параметры жизненного цикла токена.

Создание контроллера для аутентификации. Реализуется контроллер, который обрабатывает запросы на вход в систему. При успешной аутентификации пользователю возвращается JWT токен, который может быть использован для доступа к защищенным ресурсам.

**Шифрование паролей**. Пароли пользователей хранятся в зашифрованном виде, что обеспечивает их безопасность и защиту от несанкционированного доступа.

Использование ролей. В системе реализована система ролей, что позволяет ограничивать доступ к функционалу в зависимости от роли пользователя (например, администратор, редактор, пользователь).

Политики авторизации. Для определения прав доступа к определенным ресурсам используются политики, которые помогают управлять доступом на уровне контроллеров или методов.

**Использование HTTPS**. Все данные передаются по защищенному протоколу HTTPS, что обеспечивает защиту информации в процессе передачи.

**8 UNIT- И ИНТЕГРАЦИОННЫЕ ТЕСТЫ**

Для проведения юнит-тестирования был использован фреймворк *xUnit*. *xUnit* – это семейство фреймворков для автоматизированного тестирования программного обеспечения, основанных на концепции модульного тестирования. Он предоставляет набор инструментов и структур для создания и запуска тестовых сценариев.

Еще одним преимуществом *xUnit* является его расширяемость. Фреймворк предоставляет возможность создания пользовательских атрибутов, расширяющих функциональность тестовых сценариев. Это позволяет адаптировать *xUnit* под конкретные потребности проекта и упростить процесс тестирования.

Кроме того, *xUnit* обладает возможностью группировки тестовых сценариев в наборы (*test suites*) и выполнения их параллельно. Это позволяет ускорить процесс тестирования и повысить эффективность использования ресурсов.

Таблица 8.1 – Система тест-кейсов

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Заглавие тест-кейса** | **Шаги тест-кейса** | **Ожидаемый результат** |
| **UC1** | **Создание нового риска** | 1. Войти как пользователь с правами администратора. 2. Перейти в раздел «Риски». 3. Нажать «Создать риск». 4. Заполнить форму риском и сохранить. | Новый риск создается, появляется в списке рисков с актуальными данными о риске. |
| **UC2** | **Просмотр списка рисков** | 1. Войти в систему. 2. Перейти в раздел «Риски». 3. Просмотреть список рисков. | Отображается актуальный список рисков, с возможностью фильтрации по статусу, дате создания, типу риска и другим параметрам. |

Продолжение таблицы 8.1

|  |  |  |  |
| --- | --- | --- | --- |
| **UC3** | **Обновление риска** | 1. Войти как пользователь с правами администратора. 2. Перейти в раздел «Риски». 3. Выбрать риск для редактирования. 4. Изменить данные риска и сохранить. | Изменения успешно сохраняются, обновленный риск появляется в списке с актуальными данными. |
| **UC4** | **Удаление риска** | 1. Войти как пользователь с правами администратора. 2. Перейти в раздел «Риски». 3. Выбрать риск для удаления. 4. Подтвердить удаление. | Риск успешно удаляется из базы данных, его больше нет в списке рисков. |
| **UC5** | **Генерация отчета по риску** | 1. Войти как администратор. 2. Перейти в раздел «Отчеты». 3. Выбрать риск для генерации отчета. 4. Нажать «Сгенерировать отчет». | Отчет по риску успешно генерируется, и его можно просмотреть или скачать. |
| **UC6** | **Просмотр отчета по риску** | 1. Войти как администратор. 2. Перейти в раздел «Отчеты». 3. Открыть сгенерированный отчет. | Отчет по риску успешно отображается с актуальными данными. |

Окончание таблицы 8.1

|  |  |  |  |
| --- | --- | --- | --- |
| **UC7** | **Редактирование отчета по риску** | 1. Войти как администратор. 2. Перейти в раздел «Отчеты». 3. Выбрать отчет для редактирования. 4. Внести изменения и сохранить. | Изменения в отчете успешно сохраняются, и отчет обновляется с новыми данными. |
| **UC8** | **Удаление отчета по риску** | 1. Войти как администратор. 2. Перейти в раздел «Отчеты». 3. Выбрать отчет для удаления. 4. Подтвердить удаление. | Отчет по риску успешно удаляется из базы данных, его больше нет в списке отчетов. |
| **UC9** | **Просмотр подробной информации о риске** | 1. Войти в систему. 2. Перейти в раздел «Риски». 3. Выбрать риск для просмотра подробной информации. | Открывается страница с подробной информацией о риске, включая его описание, статус и связанные документы. |
| **UC10** | **Регистрация нового пользователя** | 1. Войти как администратор. 2. Перейти в раздел «Пользователи». 3. Нажать «Создать нового пользователя». 4. Заполнить данные и сохранить. | Новый пользователь успешно создается и появляется в списке пользователей. |

В результате автоматизированного тестирования функциональности ПС было подтверждено, что все тестируемые функции работают корректно и возвращают ожидаемые результаты.

Покрытие кода является метрикой, которая показывает, насколько код программы был протестирован. Оно измеряется в процентах и отражает долю выполненных строк кода или ветвей в сравнении с общим количеством строк или ветвей в коде.

**9 ОПИСАНИЕ ПРОЦЕССА РАЗВЕРТЫВАНИЯ ПС**

Для корректной работы программного средства необходимы следующие требования:

1. операционная система *Windows* 10, 11;
2. *Microsoft .NET 6 SDK*;
3. оперативная память – 2048 Мб;
4. свободное место на жестком диске – 500 Мб;
5. Docker (Docker Desktop);
6. Поддержка Linux-контейнеров в Docker.

Скрипт docker.compose:

ersion: '3.4'

services:

governmentsupportfamilyapi:

image: ${DOCKER\_REGISTRY-}governmentsupportfamilyapi

build:

context: .

dockerfile: GovernmentSupportFamilyAPI/Dockerfile

environment:

- ASPNETCORE\_ENVIRONMENT=Development

- ASPNETCORE\_URLS=https://+:443;http://+:80

ports:

- "80:80"

- "443:443"

volumes:

- ${APPDATA}/Microsoft/UserSecrets:/root/.microsoft/usersecrets:ro

- ${APPDATA}/ASP.NET/Https:/root/.aspnet/https:ro

networks:

- app-network

depends\_on:

- sql

sql:

image: "mcr.microsoft.com/mssql/server:2022-latest"

container\_name: sql\_server2022

ports:

- "1433:1433"

environment:

- ACCEPT\_EULA=y

- SA\_PASSWORD=A&VeryComplex123Password

networks:

- app-network

razorapp:

container\_name: razor\_app

build:

context: ./RazorPagesApp

dockerfile: Dockerfile

image: my-razor-app-image

ports:

- "5000:80"

networks:

- app-network

depends\_on:

- governmentsupportfamilyapi

networks:

app-network:

driver: bridge  
  
  
Выполнить в директории с образам, следующие команды:

1. docker-compose build
2. docker-compose up -d
3. docker ps

При первичном запуске серверной частей будет создана пустая база данных.

Для проверки работоспособности ПО необходимо:

* убедиться в том, что сервис базы данных запущен;
* убедиться в том, что запущена серверная часть ПС;
* запустить клиентскую часть ПС и выполнить авторизацию

# **10 РАЗРАБОТКА РУКОВОДСТВА ПОЛЬЗОВАТЕЛЯ**

Описание функций и задач, которые может выполнять разработанное программное средство представлено в таблице 10.

Таблица 10 – Описание функций и задач программного средства

|  |  |  |
| --- | --- | --- |
| Функции | Задачи | Описание |
| 1 | 2 | 3 |
| Обеспечивает управление информацией о рисках | Работа с рисками | В ходе выполнения данной задачи пользователю системы предоставляется возможность добавлять, редактировать и удалять риски, а также просматривать информацию о них. |
| Обеспечивает управление информацией о пользователях | Работа с пользователями | В ходе выполнения данной задачи пользователю системы предоставляется возможность добавлять новых пользователей, изменять их роли и удалять их. |
| Обеспечивает управление информацией о статусах заявок | Работа со статусами заявок | В ходе выполнения данной задачи пользователю системы предоставляется возможность управлять статусами заявок (создание, редактирование, удаление). |
| Обеспечивает управление информацией о категориях рисков | Работа с категориями рисков | В ходе выполнения данной задачи пользователю системы предоставляется возможность управлять категориями рисков, создавая и редактируя их. |
| Обеспечивает управление информацией о отчетах | Работа с отчетами | В ходе выполнения данной задачи пользователю системы предоставляется возможность создавать отчеты, просматривать их и экспортировать в разные форматы. |
| Обеспечивает управление информацией о статусах заявки | Работа со справочником «Статус заявки» | В ходе выполнения данной задачи пользователю системы предоставляется возможность выполнения базовых операций с данными статусах заявок |
| Обеспечивает управление информацией о заявках | Работа со справочником «Заявки» | В ходе выполнения данной задачи пользователю системы предоставляется возможность выполнения базовых операций с данными заявок |

Таблица 11 – Описание операций, реализуемых программным средством

|  |  |  |
| --- | --- | --- |
| Операция 1 | |  | | --- | | Просмотр списка рисков | |
| Условия, при соблюдении которых возможно выполнение операции | |  | | --- | | Пользователь авторизован. Пользователю доступны права. | |
| Подготовительный процесс | |  | | --- | | Не требуются | |
| Основные действия в требуемой последовательности | |  | | --- | | На странице клиентской части перейти в меню «Риски». Просмотреть все зарегистрированные риски. | |
| Заключительные действия | |  | | --- | | После завершения работы со списком рисков выйти в главное меню | |
| Ресурсы, расходуемые на операцию | |  | | --- | | 10 секунд | |
| Доступно для роли | |  | | --- | | Аудитор, Руководитель | |
| Операция 2 | Создание нового риска |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Риски». Нажать «Создать риск». |
| Основные действия в требуемой последовательности | Заполнить форму для нового риска и сохранить. |
| Заключительные действия | После сохранения новый риск появляется в списке рисков. |

Продолжение таблицы 11

|  |  |
| --- | --- |
| Ресурсы, расходуемые на операцию | 15 секунд |
| Доступно для роли | Аудитор |
| Операция 3 | Обновление риска |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Риски». |
| Основные действия в требуемой последовательности | Выбрать риск для редактирования. Изменить данные риска и сохранить. |
| Заключительные действия | Изменения сохраняются, обновленный риск появляется в списке. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Аудитор |
| Операция 4 | Удаление риска |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Риски». |
| Основные действия в требуемой последовательности | Выбрать риск для удаления. Подтвердить удаление. |
| Заключительные действия | Риск удаляется из базы данных, больше не отображается в списке. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Аудитор |
| Операция 5 | Генерация отчета по риску |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |

Продолжение таблицы 11

|  |  |
| --- | --- |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Отчеты». |
| Основные действия в требуемой последовательности | Выбрать риск для генерации отчета. Нажать «Сгенерировать отчет». |
| Заключительные действия | Отчет успешно генерируется, его можно просмотреть или скачать. |
| Ресурсы, расходуемые на операцию | 20 секунд |
| Доступно для роли | Аудитор, Руководитель |
| Операция 6 | Просмотр отчета по риску |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Отчеты». |
| Основные действия в требуемой последовательности | Открыть сгенерированный отчет. |
| Заключительные действия | Отчет отображается с актуальными данными. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Аудитор, Руководитель |
| Операция 7 | Редактирование отчета по риску |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Отчеты». |
| Основные действия в требуемой последовательности | Выбрать отчет для редактирования. Внести изменения и сохранить. |
| Заключительные действия | Отчет обновляется с новыми данными. |
| Ресурсы, расходуемые на операцию | 15 секунд |

Продолжение таблицы 11

|  |  |
| --- | --- |
| Доступно для роли | Аудитор |
| Операция 8 | Удаление отчета по риску |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Отчеты». |
| Основные действия в требуемой последовательности | Выбрать отчет для удаления. Подтвердить удаление. |
| Заключительные действия | Отчет удаляется из базы данных. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Руководитель |
| Операция 8 | Регистрация нового пользователя |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Пользователи». |
| Основные действия в требуемой последовательности | Нажать «Создать нового пользователя». Заполнить данные и сохранить. |
| Заключительные действия | Новый пользователь успешно создается и появляется в списке пользователей. |
| Ресурсы, расходуемые на операцию | 15 секунд |
| Доступно для роли | Администратор |
| Операция | Удаление пользователя |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Пользователи». Выбрать пользователя для удаления. |

Окончание таблицы 11

|  |  |
| --- | --- |
| Заключительные действия | Пользователь успешно удаляется из системы, его больше нет в списке пользователей. Отобразится сообщение о том, что операция завершена успешно. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Администратор |
| Операция | Редактирование пользователя |
| Условия, при соблюдении которых возможно выполнение операции | Пользователь авторизован. Пользователю доступны права администратора. |
| Подготовительный процесс | На странице клиентской части перейти в раздел «Пользователи». Выбрать пользователя для редактирования. |
| Основные действия в требуемой последовательности | Выбрать пользователя из списка, нажать кнопку «Редактировать». Изменить нужные данные (например, имя, email, роль) и сохранить изменения. |
| Заключительные действия | Изменения успешно сохраняются, и обновленный пользователь появляется в списке. Отобразится сообщение о том, что операция завершена успешно. |
| Ресурсы, расходуемые на операцию | 10 секунд |
| Доступно для роли | Администратор |

Раздел описывает ключевые задачи и операции системы управления контрагентами, детализируя условия их выполнения, последовательность действий и временные затраты. Каждая задача четко структурирована и включает информацию о необходимых ресурсах, что позволяет пользователю легко понять логику работы системы и ее функциональные возможности. Простота использования. Все задачи имеют четкую последовательность действий, минимум подготовительных шагов и интуитивно понятный интерфейс.

# **ВЫВОД**

Документ представляет собой подробный отчет о разработке программного средства для управления корпоративными рисками на основе данных внутреннего аудита. Проект демонстрирует высокий профессионализм и системный подход к созданию программного обеспечения.

Система построена на современной трехуровневой архитектуре, которая эффективно разделяет функциональные компоненты, обеспечивая гибкость и масштабируемость приложения. Интерфейс был разработан с учетом потребностей пользователей, таких как аудиторы и руководители, при этом сохраняя единый стиль и интуитивно понятную навигацию. Каждый из типов пользователей имеет доступ к специфическим функциям системы, что улучшает эффективность работы и принятие решений.

Визуальное оформление системы продумано с акцентом на легкость восприятия, что особенно важно для аудиторов, которым необходимо быстро и точно обрабатывать данные, а также для руководителей, которым нужны четкие аналитические отчеты. Тщательно подобранная цветовая гамма и грамотная типографика способствуют комфортному взаимодействию с системой, обеспечивая четкость и внимание к важной информации.

База данных спроектирована с соблюдением принципов нормализации до третьей нормальной формы (3НФ), что позволяет обеспечить высокую целостность данных и оптимизировать их обработку. Для обеспечения безопасности системы применяются современные механизмы аутентификации на базе JWT-токенов, а также надежные методы хеширования паролей, что предотвращает утечку данных и несанкционированный доступ.

Процесс развертывания системы полностью документирован, и руководство пользователя содержит все необходимые инструкции для эффективной работы с приложением. В результате, разработанная система представляет собой масштабируемое и безопасное решение, которое объединяет удобство использования с мощным аналитическим функционалом.