**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 7](#_Toc194309834)

[1. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 8](#_Toc194309835)

[1.1 Обоснование выбора средств реализации компонента(ов) архитектурного решения 8](#_Toc194309836)

[1.1.1 Сравнительный обзор современных средств разработки клиентской части 9](#_Toc194309837)

[1.1.2. Сравнительный обзор современных СУБД 11](#_Toc194309838)

[1.2. Описание реализации компонента(ов) архитектурного решения 13](#_Toc194309839)

[1.3. Обеспечение информационной безопасности архитектурного решения 23](#_Toc194309840)

[1.4. Описание эксплуатации компонентов архитектурного решения 24](#_Toc194309841)

[ЗАКЛЮЧЕНИЕ 27](#_Toc194309842)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc194309843)

# ВВЕДЕНИЕ

# 1. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

## 1.1 Обоснование выбора средств реализации компонента(ов) архитектурного решения

При разработке веб-приложения важно учитывать надежность, производительность, удобство разработки и поддержки кода. В данном проекте были выбраны следующие ключевые инструменты:

1. Язык программирования — Python.
2. Фреймворк — Django.
3. Среда разработки — PyCharm.
4. Система управления базами данных (СУБД) — PostgreSQL.

Эти технологии обеспечивают гибкость, удобство разработки, поддержку масштабируемости и безопасности, что делает их оптимальным решением для создания веб-приложения. Рассмотрим каждую из этих технологий подробнее.

#### Таблица 1.1 — Сравнительная характеристика среды разработки

|  |  |  |
| --- | --- | --- |
| Среда разработки | Преимущества | Недостатки |
| PyCharm | Интеллектуальная поддержка Python, встроенные инструменты для Django и работы с базами данных, удобный отладчик | Высокое потребление ресурсов, платная профессиональная версия |
| Visual Studio Code | Легкость, множество расширений, бесплатность | Ограниченный встроенный функционал для Python |
| Eclipse (с PyDev) | Мощный инструмент, поддержка различных языков | Сложная настройка, перегруженный интерфейс |
| Atom | Простота, кроссплатформенность | Медленная работа с большими проектами |

Для реализации проекта выбрана среда разработки **PyCharm**, так как она предоставляет удобные инструменты для работы с кодом, отладкой и тестированием. PyCharm поддерживает интеграцию с Django, PostgreSQL и другими необходимыми инструментами, что делает разработку более эффективной. Кроме того, среда обладает интеллектуальной подсветкой синтаксиса, автодополнением кода и мощными инструментами анализа, что значительно упрощает процесс написания программного кода.

### 1.1.1 Сравнительный обзор современных средств разработки клиентской части

Для разработки клиентской части веб-приложений используется множество различных инструментов и технологий. В качестве средств разработки клиентской части могут быть рассмотрены:

1. Python с фреймворком Django:

Преимуществами комбинации являются:

* простота синтаксиса, что делает Python доступным для новичков;
* высокая производительность благодаря интеграции с PostgreSQL;
* сильное сообщество и поддержка документации.

Недостатки:

* потребность в серверной части, так как Python в основном используется для серверных решений.

Таким образом, Django предоставляет встроенные возможности для разработки интерфейсов, маршрутизации и работы с базами данных. Отличается удобством и возможностями расширения с помощью сторонних библиотек.

1. JavaScript (с использованием фреймворков или без них):

Преимуществами комбинации являются:

* множество библиотек и фреймворков, таких как React или Vue.js, для создания динамичных и интерактивных интерфейсов;
* нативная поддержка во всех современных браузерах;
* широкое сообщество и множество образовательных ресурсов.

Недостатки:

* для сложных приложений может потребоваться настройка серверной части для обработки данных и повышения производительности;
* в более сложных случаях может возникнуть необходимость в использовании дополнительных инструментов, таких как сборщики и транспилеры.

Таким образом, выбор JavaScript для клиентской части обеспечивает гибкость и совместимость, позволяя создавать интерактивные веб-приложения.

1. TypeScript с фреймворком Angular:

Преимуществами комбинации являются:

* Статическая типизация позволяет легко масштабировать проекты и повышает надежность кода;
* Большое сообщество и документация.

Недостатки:

* Высокая сложность в освоении по сравнению с JavaScript;
* Потребность в настройке компиляции и дополнительных инструментах.

Таким образом, Angular является фреймворком с большим количеством встроенных инструментов для создания сложных клиентских приложений.

Выбор **Python** в качестве основного языка программирования обусловлен его простым и читаемым синтаксисом, что ускоряет процесс разработки и снижает вероятность ошибок. Python обладает обширной экосистемой библиотек и фреймворков, что делает его универсальным инструментом для создания веб-приложений. Кроме того, активное сообщество разработчиков и широкая документация позволяют оперативно находить решения возникающих проблем.

Использование **Django** в качестве фреймворка для разработки веб-приложения оправдано его встроенными инструментами, обеспечивающими быструю и удобную разработку. Django предлагает мощную систему аутентификации, автоматическое управление базой данных, гибкую систему маршрутизации и высокий уровень безопасности. Это делает его отличным выбором для создания масштабируемых и надежных веб-приложений.

Фронтенд веб-приложения будет реализован с использованием JavaScript, что обеспечит динамичность и интерактивность интерфейса. JavaScript является стандартом для разработки клиентской части, и его широкая поддержка в браузерах позволяет гарантировать совместимость и производительность. В дополнение к этому, можно использовать различные библиотеки и фреймворки для повышения эффективности разработки.

### 1**.1.2. Сравнительный обзор современных СУБД**

Современные системы управления базами данных (СУБД) предоставляют разнообразные возможности для хранения и обработки данных. Рассмотрим несколько популярных СУБД.

1. PostgreSQL:

Преимущества:

* высокая производительность и масштабируемость;
* поддержка сложных запросов и транзакций;
* полная совместимость с SQL;
* бесплатность и открытый исходный код.

Недостатки:

* может быть сложнее в настройке и обслуживании по сравнению с другими СУБД.

Таким образом, PostgreSQL идеально подходит для сложных систем, где необходимо хранение больших объемов данных, высокая производительность и отказоустойчивость.

1. MySQL

Преимущества:

* простота использования и большая популярность;
* подходит для приложений с менее сложной структурой данных;
* хорошая производительность при высоких нагрузках.

Недостатки:

* ограниченные возможности при работе с транзакциями и сложными запросами.

Таким образом, MySQL можно выбрать для менее сложных проектов, однако она уступает PostgreSQL по возможностям работы с транзакциями и сложными запросами.

1. SQLite:

Преимущества:

* легковесность и простота использования;
* подходит для небольших проектов и приложений с низким уровнем нагрузки;
* быстрое встраивание в приложение.

Недостатки:

* не предназначена для работы с большими объемами данных и высокими нагрузками.

Таким образом, SQLite идеален для небольших приложений и прототипов, но для больших и масштабируемых проектов лучше выбрать PostgreSQL.

1. Oracle:

Преимущества:

* высокая производительность и надежность;
* большие возможности для работы с бизнес-данными.

Недостатки:

* высокая стоимость лицензий;
* сложность в настройке и обслуживании.

Таким образом, Oracle является отличным решением для крупных корпоративных приложений, однако из-за высокой стоимости лицензий и сложности администрирования она не является лучшим выбором для малых и средних проектов.

В качестве системы управления базами данных выбрана **PostgreSQL**, так как она обеспечивает высокую производительность, надежность и поддержку сложных SQL-запросов. PostgreSQL отличается хорошей масштабируемостью, что позволяет эффективно работать с большими объемами данных. Открытый исходный код и активное сообщество обеспечивают постоянное развитие и улучшение данной СУБД, что делает её лучшим выбором для данного проекта.

## **1.2. Описание реализации компонента(ов) архитектурного решения**

### 1.2.1 Описание реализации клиентской части ИС

Разработка компонентов клиентской части информационной системы начинается с построения дерева функций, сценария взаимодействия с пользователем и структурной схемы.

На первом этапе формируется дерево функций: определяются ключевые функции системы, которые затем детализируются до отдельных пользовательских операций.

Основные функции информационной системы:

1. Управление договорами (просмотр, добавление, редактирование, удаление).
2. Отслеживание этапов выполнения (фиксация статусов, расчет прогресса).
3. Генерация отчетов (автоматическая генерация, экспорт).
4. Работа с документами (загрузка, хранение).
5. Управление пользователями (авторизация, назначение ролей).

На Рисунке 1.1 представлено дерево функций и их декомпозиция.

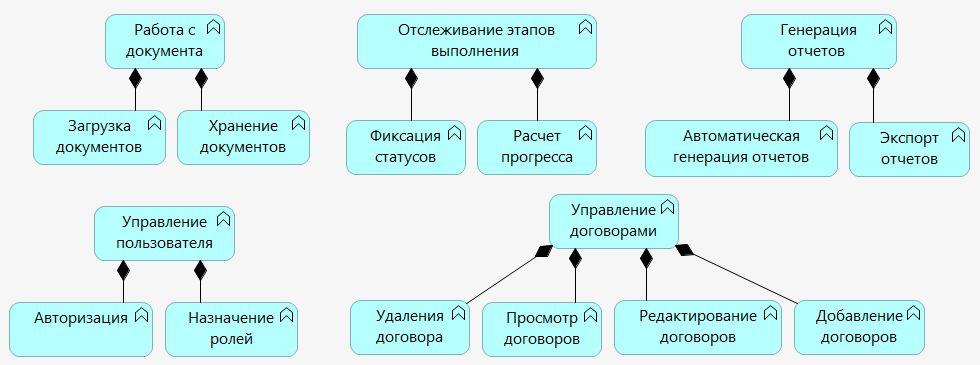


Рисунок 1.1 – Дерево функций

В результате были выделены пользовательские операции, необходимые для реализации каждой из основных функций системы. Эти операции охватывают взаимодействие пользователя с интерфейсом для достижения конкретных целей.

Служебные функции не включаются в дерево функций, поскольку не представляют собой непосредственные действия пользователя. Они предназначены для обеспечения общего функционирования системы, поддержки работоспособности и безопасности, но они не связаны напрямую с основными задачами, выполняемыми пользователем.

Служебные функции информационной системы:

1. Аутентификация и авторизация.
2. Шифрование данных.
3. Резервное копирование.
4. Мониторинг производительности.

Сценарий диалога для руководителя проекта представлен на Рисунке 1.2.

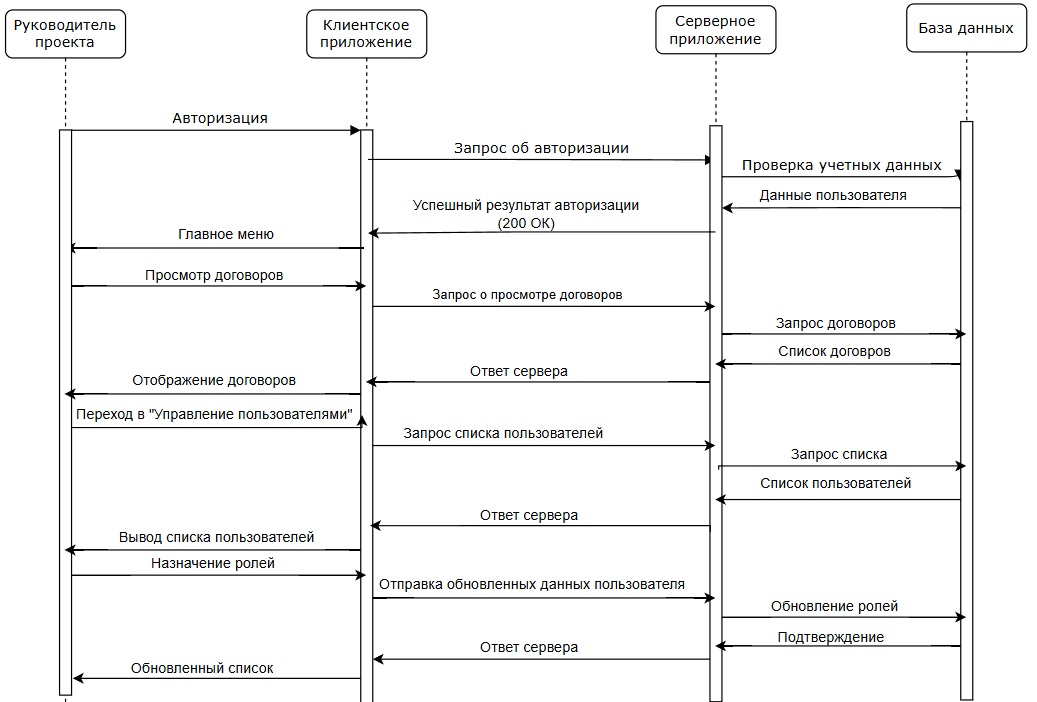


Рисунок 1.2 - Диаграмма последовательности руководителя проекта

Сценарий диалога для сотрудника бэк-офиса представлен на Рисунке 1.3.

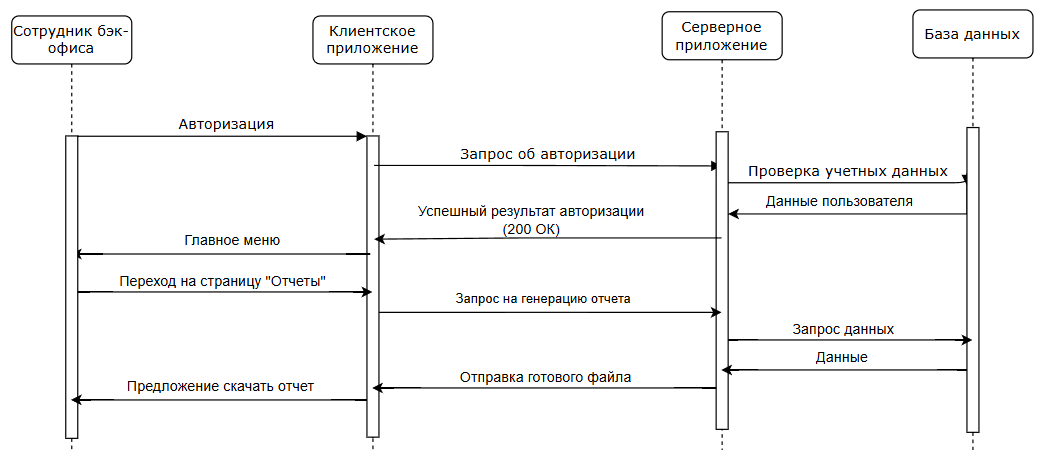


Рисунок 1.3 - Диаграмма последовательности сотрудника бэк-офиса

Сценарий диалога для менеджера проекта представлен на Рисунке 1.4.

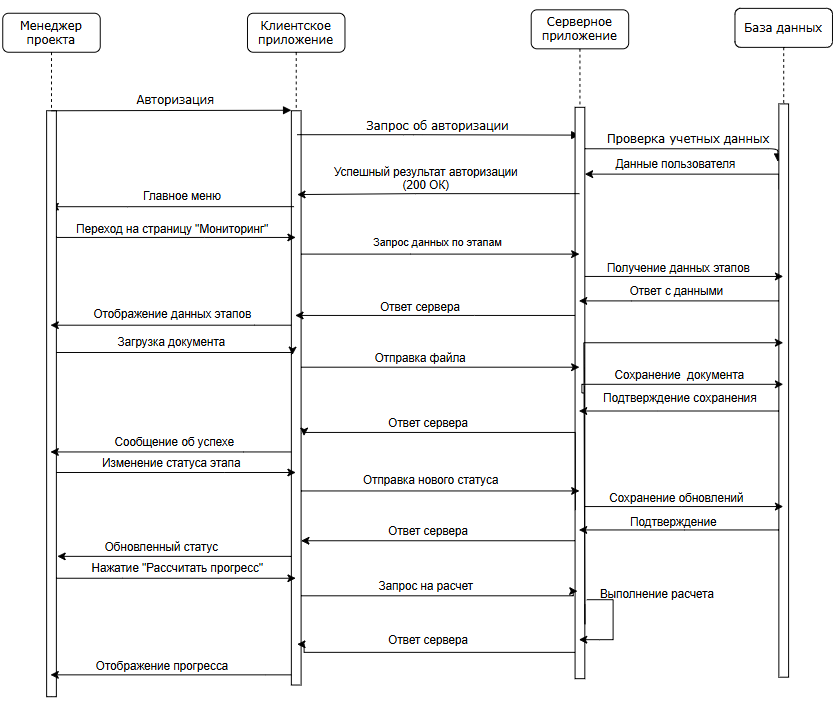


Рисунок 1.4 - Диаграмма последовательности сотрудника менеджера проекта

Для разработки структурной схемы системы, исходя из ранее описанных функций пользователей и потоков входной/выходной информации, определены следующие модули системы и их функциональные возможности.

Модуль управления пользователями:

* авторизация пользователей по логину и паролю;
* регистрация новых пользователей.

Модуль управления документами:

* загрузка и удаление файлов, связанных с проектами;
* хранение и просмотр списка документов по проектам;
* предоставление доступа к документам в соответствии с ролями пользователей.

Модуль управления проектами и мониторинга:

* просмотр информации по текущим проектам и этапам;
* инициация мониторинга выполнения проектов;
* отображение текущего состояния выполнения этапов;
* расчет и отображение общего прогресса выполнения проекта.

Модуль управления этапами проекта:

* ввод и редактирование этапов;
* фиксация выполнения этапов проекта;
* отслеживание сроков и статуса выполнения.

Модуль анализа выполнения проекта:

* расчет прогресса выполнения на основе завершенных этапов;
* визуализация прогресса в процентах.

Модуль отчетности:

* формирование отчетов по проектам, этапам, документам;
* генерация отчетов в форматах PDF;
* доступ к отчетам для сотрудников бэк-офиса и руководителей.

Структурная схема представлена на Рисунке 1.5.

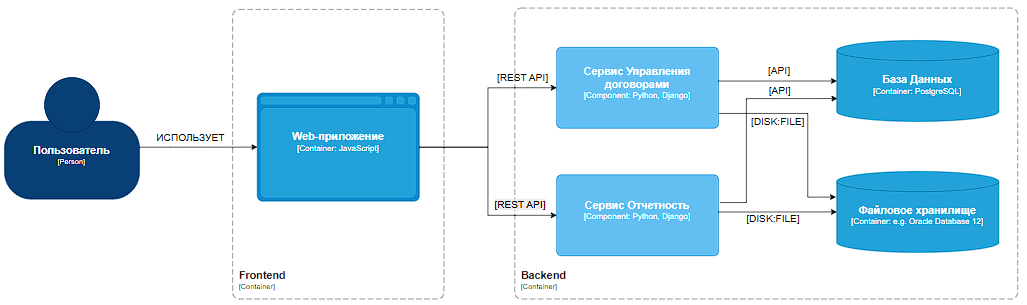


Рисунок 1.5 - Структурная схема компонентов информационной системы

Информационная система реализована по архитектуре **«клиент–сервер»** с использованием шаблона проектирования **MVC (Model–View–Controller) .**Все модули взаимодействуют друг с другом по протоколу **HTTP**. Передача данных между модулями осуществляется в формате **JSON**, что обеспечивает структурированность и удобство обработки. Клиентская часть системы реализована с использованием JavaScript. Веб-интерфейс формируется динамически на стороне клиента, а данные получаются от сервера через HTTP-запросы (в формате JSON).

Клиентская часть реализована с использованием JavaScript и стандартных веб-технологий (HTML, CSS). Взаимодействие с сервером осуществляется через REST API. Полученные от сервера данные отображаются динамически в интерфейсе. Пользователь взаимодействует с системой через удобный веб-интерфейс, доступный в зависимости от его роли.

В Таблице 1.2 приведены логические модули сервиса «Управление договорами».

Таблица 1.2 – Логические модули сервиса «Управление договорами»

|  |  |
| --- | --- |
| Модуль | Назначение |
| UserManagementModule | Управление пользователями: регистрация, авторизация, разграничение ролей |
| ContractManagementModule | Работа с договорами: создание, редактирование, загрузка файлов |
| StageTrackingModule | Ведение этапов исполнения договоров: фиксация статусов, сроков, событий |
| ProgressCalculationModule | Автоматизированный расчёт прогресса выполнения договора |

В Таблице 1.3 приведены логические модули сервиса «Отчётность»

Таблица 1.3 – Логические модули сервиса «Отчётность»

|  |  |
| --- | --- |
| Модуль | Назначение |
| ReportingModule | Генерация отчётов по исполнению договоров и их этапов |
| MonitoringModule | Централизованный мониторинг текущего состояния исполнения обязательств |

Все модули взаимодействуют с базой данных **PostgreSQL** через встроенную ORM Django. Связь с базой осуществляется через **внутренний API**, данные хранятся в нормализованной структуре, отражающей модель предметной области.

Форматы передачи данных:

* между клиентом и сервером: HTTP / JSON;
* между сервером и базой данных: SQL-запросы через ORM Django.

### 1.2.2 Описание реализации базы данных ИС

Физическая модель данных информационной системы реализована с использованием реляционной СУБД **PostgreSQL**, которая обеспечивает надежное хранение, обработку и обеспечение целостности данных. Модель данных основана на логической структуре системы, описанной ранее, и реализована в виде набора взаимосвязанных таблиц.

Основные таблицы базы данных приведены ниже.

Таблица 1.4 описывает таблицу пользователей системы.

#### Таблица 1.4 — Пользователь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | Обязательность | Значение по умолчанию | Формат / проверка |
| id | SERIAL | + | Автоинкремент | Первичный ключ |
| Username | VARCHAR(150) | + | - | Уникальность |
| Password | VARCHAR(128) | + | - | Хеш пароля |
| role | VARCHAR(50) | + | 'employee' | CHECK (manager / admin / analyst) |
| email | VARCHAR(255) | - | NULL | Формат email |

Таблица 1.5 описывает таблицу проектов.

#### Таблица 1.5 — Проекты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | Обязательность | Значение по умолчанию | Связь |
| id | SERIAL | + | Автоинкремент | Первичный ключ |
| Username | VARCHAR(150) | + | - | |  | | --- | |  | |
| description | TEXT | - | NULL |  |
| created\_at | TIMESTAMP | + | Now() |  |
| manager\_id | INTEGER | + | - | |  | | --- | | FK – users(id) | |

Таблица 1.6 описывает таблицу этапов выполнения проектов.

#### Таблица 1.6 — Этапы выполнения проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | Обязательность | Значение по умолчанию | Связь |
| id | SERIAL | + | - | Первичный ключ |
| project\_id | INTEGER | + | - | |  | | --- | |  |  FK – projects(id) |
| name | VARCHAR(255) | + | - |  |
| status | VARCHAR(50) | + | 'not\_started' |  |
| start\_date | |  | | --- | |  |  |  | | --- | | DATE | | - | |  | | --- | |  |  |  | | --- | | NULL | | |  |  | | --- | --- | |  |  |  CHECK (not\_started, in\_progress, done) |
| end\_date | |  | | --- | |  |  |  | | --- | | DATE | | - | |  | | --- | |  |  |  | | --- | | NULL | |  |

Таблица 1.7 описывает таблицу документов.

#### Таблица 1.7 — Документы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | Обязательность | Значение по умолчанию | Связь |
| id | SERIAL | + | - | Первичный ключ |
| file\_path | VARCHAR(255) | + | - | |  | | --- | |  | |
| project\_id | INTEGER | + | - | FK - projects(id) |
| uploaded\_at | TIMESTAMP | + | now() |  |

Таблица 1.8 описывает таблицу отчетов.

#### Таблица 1.8 — Отчеты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | Обязательность | Значение по умолчанию | Связь |
| id | SERIAL | + | - | Первичный ключ |
| project\_id | INTEGER | + | - | |  | | --- | |  |  FK - projects(id) |
| generated\_by | INTEGER | + | - | FK - users(id) |
| created\_at | TIMESTAMP | + | now() |  |
| content | |  | | --- | |  |  TEXT | + | |  | | --- | |  |  - | |  |  | | --- | --- | |  |  | |

Связи между таблицами:

Один пользователь может быть менеджером нескольких проектов. Каждый проект может включать несколько этапов. К каждому проекту могут быть прикреплены документы. Отчеты генерируются по проекту и связаны с пользователем.

На Рисунке 1.6 представлена физическая модель базы данных.

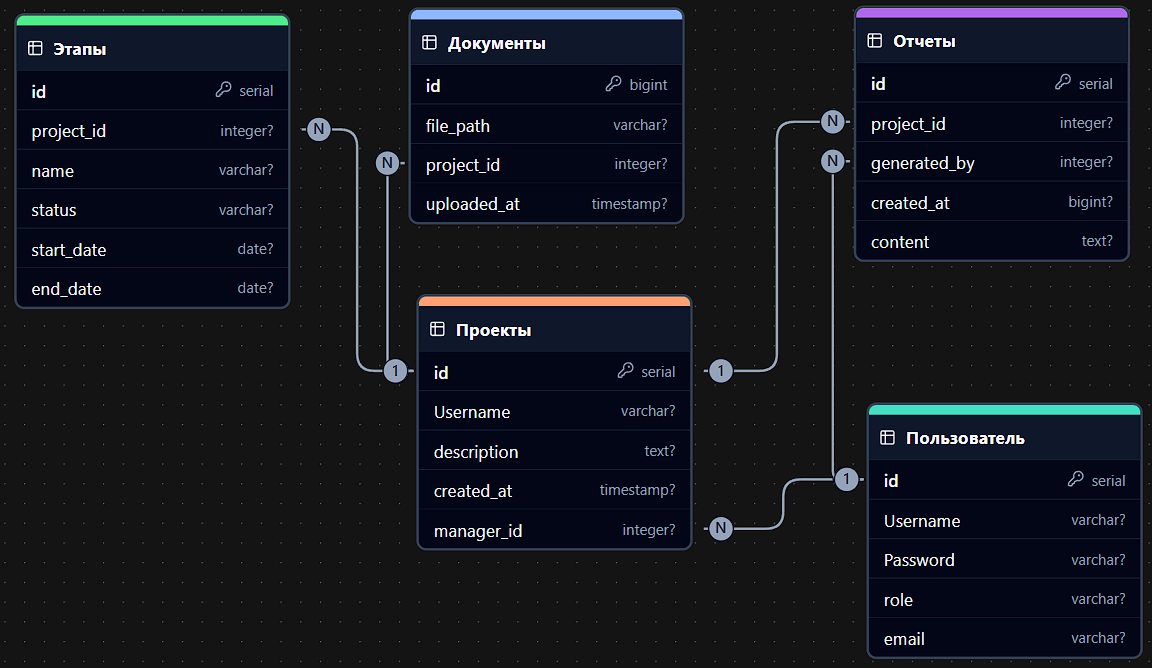


Рисунок 1.6 - Физическая модель базы данных

Для формирования отчетов и отображения данных на веб-страницах используются SQL-запросы, встроенные в Django ORM.

Пример SQL-запроса для предоставления списка проектов с менеджерами представлен в Листинге 1.1.

Листинг 1.1 – SQL-запрос для предоставления списка проектов с менеджерами

|  |
| --- |
| SELECTp.name AS project\_name,u.username AS manager\_name,p.created\_at::date AS creation\_dateFROMprojects pJOINusers u ON p.manager\_id = u.idORDER BYp.created\_at DESC; |

Серверная часть разделена на два ключевых логических сервиса - сервис «Управление договорами», сервис «Отчётность».

В Таблице 1.9 представлены **логические и физические идентификаторы модулей** информационной системы мониторинга исполнения договоров. Она поможет связать структуру ИС с деревом функций и архитектурой проекта.

Таблица 1.9 - логические и физические идентификаторы модулей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Логическое название модуля | Физическое название (идентификатор в Django) | Основные реализуемые функции |
| 1 | Модуль управления пользователями | Users | Регистрация, авторизация, разграничение прав доступа |
| 2 | Модуль управления договорами | Contracts | Создание, редактирование договоров, загрузка файлов |
| 3 | Модуль отслеживания этапов исполнения | Stages | Ввод информации об этапах, изменение статусов |
| 4 | Модуль расчёта прогресса исполнения | Progress | Расчёт процента выполнения этапов и договора в целом |
| 5 | Модуль мониторинга состояния договоров | Monitoring | |  | | --- | |  |  |  | | --- | | Отображение текущего состояния исполнения всех договоров | |
| 6 | Модуль генерации отчётов | Reporting | Формирование текстовых и табличных отчётов по этапам и договорам |

Эти модули организованы в виде приложений Django и подключаются в основном файле проекта через INSTALLED\_APPS. Внутри каждого приложения реализуются представления (views.py), модели (models.py) отвечающие за соответствующую функциональность.

## 1.3. Обеспечение информационной безопасности архитектурного решения

Для обеспечения достоверности информации в системе реализован механизм криптографического хеширования с использованием SHA-256. Это позволяет фиксировать любые изменения в данных о договорах, этапах их выполнения и финансовых показателях. Каждая операция модификации данных сопровождается автоматической проверкой контрольных сумм. В системе имеется многоуровневая система аутентификации.

Доступ к системе организован по принципу минимально необходимых привилегий:

* все пользователи проходят обязательную двухэтапную проверку (логин/пароль + SMS-код);
* для администраторов предусмотрены аппаратные ключи доступа.

Разграничение прав реализовано через детальную систему ролей, где для каждой должности четко определен перечень доступных операций.

Система обеспечивает надежное хранение информации за счет:

* ежечасного резервного копирования измененных данных;
* полного копирования базы данных раз в сутки;
* хранения архивных копий в географически распределенном хранилище.

Процедура восстановления позволяет вернуть систему в рабочее состояние не позднее чем через 20 минут после сбоя.

Для предотвращения утечек конфиденциальной информации применяются:

* сквозное шифрование всех передаваемых данных;
* шифрование хранимой информации с использованием AES-256;
* автоматическое скрытие конфиденциальных данных в интерфейсе.

Архитектура системы разработана с учетом требований Федерального закона №152-ФЗ, отраслевых стандартов информационной безопасности, внутренних регламентов компании. Также регулярно проводятся независимые аудиты безопасности.

Для оперативного выявления угроз реализованы:

* круглосуточный контроль активности пользователей;
* автоматическое блокирование подозрительных действий;
* регулярное тестирование на уязвимости.

Комплексный подход к безопасности обеспечивает надежную защиту от всех основных видов кибератак.

Примененные решения гарантируют сохранность данных о договорах, предотвращают несанкционированный доступ и обеспечивают стабильную работу системы. Это позволяет минимизировать операционные риски и поддерживать непрерывность бизнес-процессов компании.

## 1.4. Описание эксплуатации компонентов архитектурного решения

Разработанная информационная система, предназначенная для мониторинга исполнения договоров, построена на клиент-серверной архитектуре. Для её запуска и повседневной работы пользователю не требуется специальное программное обеспечение — достаточно стандартного веб-браузера и доступа к сети Интернет.

Работа с системой начинается просто: пользователь открывает браузер, вводит в адресной строке URL, выданный администратором (https://monitoring.sila-union.ru), и попадает на страницу авторизации. На экране отображается привычная форма входа, где нужно ввести логин и пароль.

После успешной авторизации пользователь приветствуется системой и перенаправляется на главную страницу. Здесь он сразу видит дашборд с основными данными: список текущих договоров, стадии их выполнения и доступные отчёты. Если ввод был некорректным, система выдаст ошибку: «Повторите попытку».

Раздел Договоры служит основным рабочим пространством для менеджера. Здесь можно добавить новый договор: достаточно нажать кнопку «Добавить Договор», заполнить базовую информацию, выбрать ответственного сотрудника и прикрепить текст договора. После сохранения он появляется в списке, а система подтверждает действие сообщением «Договор успешно создан». Все это оформлено в виде простой и наглядной формы. Также представлен общий список договоров.

При переходе в карточку конкретного договора пользователь попадает на вкладку Этапы, где отображаются все этапы реализации. Добавление нового этапа происходит через интуитивно понятную форму: указываются наименование, сроки и текущий статус. Система автоматически рассчитывает прогресс выполнения договора и отображает его в процентах. Таким образом, и менеджер, и руководитель могут быстро оценить текущее положение дел.

Загрузить дополнительные материалы — сканы, акты или таблицы — можно во вкладке Документы. Поддерживаются популярные форматы, включая PDF, DOCX и XLSX. Загрузка выполняется в несколько шагов: выбор файла, привязка к нужному договору, сохранение. После этого документ становится доступен для просмотра.

Для генерации отчётов предусмотрен отдельный раздел Отчеты. Сотрудник бэк-офиса или руководитель выбирает интересующий проект или временной диапазон, после чего система формирует отчёт в формате PDF. Как только отчёт готов, появляется сообщение: «Отчет успешно сгенерирован».

Пользователи системы различаются по ролям: менеджер, руководитель проекта, сотрудник бэк-офиса. Только руководитель проекта имеет доступ к разделу Пользователи, где можно добавлять новых сотрудников. Процесс включает ввод логина, пароля и выбор роли. Система автоматически применяет соответствующие права доступа.

Представленная система демонстрирует продуманную структуру и логичную последовательность действий. Благодаря понятному интерфейсу и сопровождающим подсказкам, обучение новым пользователям занимает минимум времени. Использование привычных веб-технологий делает систему доступной с любого устройства и операционной системы, а разграничение ролей помогает организовать чёткое распределение обязанностей.

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.