МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка социальных и экономических информационных систем
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка информационной системы отслеживания документов для ЦЦК

Обучающийся Д.Н. Карамалишоева _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд.пед.наук, доцент, О.М. Гущина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд.пед.наук, доцент, А.В.Егоровна
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Название бакалаврской работы – «Разработка информационной системы отслеживания документов для ЦЦК».

Объект исследования бакалаврской работы является процесс хранения, изменения статуса и манипулирования данными договоров и актов. Предмет исследования является разработка и внедрение информационной системы отслеживания статуса документов для улучшения управления документами в ЦЦК.

Цель работы: разработать и внедрить информационную систему отслеживания статуса документов, которая позволит улучшить управление документами в Центре цифровых компетенций (ЦЦК), обеспечив их эффективное хранение, доступность и отслеживание.

Для достижения необходимой цели нужно выполнить такие задачи как:

- анализ текущей ситуации: Исследование и анализ существующих методов управления документами в ЦЦК, выявление их недостатков и проблем;
- исследование и анализ требований: определение основных функциональных требований к системе, анализ потребностей пользователей и организации, исследование существующих решений и технологий;
- разработка архитектуры системы: определение структуры базы данных для хранения документов, разработка архитектуры пользовательского интерфейса, определение модулей системы и их взаимодействия;
- разработка пользовательского интерфейса: реализация функционала для просмотра, добавления, редактирования и удаления документов, разработка функционала поиска и фильтрации документов;
- тестирование и оптимизация системы: проведение функционального и юзабилити тестирования для проверки корректности работы всех

модулей системы, тестирование производительности системы под нагрузкой, оптимизация системы для улучшения скорости работы и эффективности использования ресурсов.

Бакалаврская работа состоит из введения, трех глав и заключения.

В первой главе мы рассмотрим фундаментальные принципы и концепции, лежащие в основе информационных систем, а также определим ключевые требования и цели, которые должна удовлетворять наша система.

Во второй главе описывается процесс проектирования системы, включая выбор архитектуры, определение требований к функциональности и интерфейсам, а также разработку схемы базы данных.

В третьей главе описывается процесс реализации и тестирование ИС.

Бакалаврская работа содержит пояснительную записку объемом 60 страниц, включает в себя 16 рисунков, список использованной литературы, состоящий из 25 источника.

Abstract

The title of the bachelor's thesis is "Development of an information system for tracking documents for the CCC".

The object of research of the bachelor's work is the process of storing, changing the status and manipulating data of contracts and acts.

The subject of the study is the development and implementation of an information system for tracking the status of documents to improve document management in the CCC.

The purpose of the work: to develop and implement an information system for tracking the status of documents, which will improve document management in the Center for Digital Competencies (CCC), ensuring their effective storage, accessibility and tracking.

To achieve the necessary goal, you need to complete tasks such as:

- analysis of the current situation: Research and analysis of existing document management methods in the CEC, identification of their shortcomings and problems;
- requirement’s research and analysis: identification of the main functional requirements for the system, analysis of user and organization needs, research of existing solutions and technologies;
- development of the system architecture: definition of the database structure for storing documents, development of the user interface architecture, definition of the system modules and their interaction;
- user interface development: implementation of functionality for viewing, adding, editing and deleting documents, development of functionality for searching and filtering documents;
- testing and optimization of the system: conducting functional and usability testing to verify the correct operation of all modules of the system, testing system performance under load, optimizing the system to improve the speed and efficiency of resource use.

The bachelor's thesis consists of an introduction, three chapters and a conclusion.

In the first chapter, we will look at the fundamental principles and concepts underlying information systems, as well as identify the key requirements and goals that our system must meet.

The second chapter describes the system design process, including the choice of architecture, defining requirements for functionality and interfaces, and developing a database schema.

The third chapter describes the process of implementing and testing the IP.

Does the bachelor's thesis contain an explanatory note in volume 60 pages, does it include 16 drawings and a list of references, consisting of 25 the source.

Оглавление

Введение 7
Глава 1 Теоретические основы ИС и анализ требований 9
1.1 Общие сведения о Тольяттинском государственном университете 9
1.2 Общие сведения о ЦЦК 10
1.3 Классификация и принцип работы информационных систем 14
1.4 Анализ требований к системе отслеживания статуса документов ЦЦК 20
Глава 2 Проектирование, технологические решения и выборы для информационной системы отслеживания документов ЦЦК 23
2.1 Выбор архитектурного подхода 23
2.2 Выбор Django как фреймворка для разработки 26
2.3 Выбор СУБД для хранения данных 28
2.4 Определение структуры базы данных для отслеживания статуса документов ЦЦК 30
Глава 3 Разработка и тестирование информационной системы отслеживания статуса документов ЦЦК 36
3.1 Реализация информационной системы отслеживания документов 36
3.2 Тестирование информационной системы отслеживания документов 43
Заключение 52
Список используемой литературы 53
Приложение А Фрагмент кода создание сущностей 55
Приложение Б Фрагмент кода конфигурации административной панели 58

Введение

В рамках данной ВКР рассматривается разработка информационной системы отслеживания статуса документов ЦЦК, цель которой - обеспечить эффективное управление документами в организации, повысить их доступность и улучшить качество работы. В условиях цифровизации и роста объемов информации, управление документами становится все более сложным и многоуровневым.

Разработка информационной системы отслеживания статуса документов для ЦЦК представляет собой важный шаг в направлении цифровой трансформации образовательного процесса. Эта система позволит автоматизировать процесс управления документами, обеспечивая их эффективное хранение, доступность и отслеживание, что в свою очередь повысит эффективность работы ЦЦК и улучшит качество образовательного процесса.

Актуальность данной темы обусловлена необходимостью внедрения современных технологий в образовательный процесс, а также потребностью в улучшении управления документами в ЦЦК для повышения его эффективности.

Объект исследований бакалаврской работы является процесс хранения, изменения статуса и манипулирования данными договоров и актов.

Предмет исследования является разработка и внедрение информационной системы отслеживания статуса документов для улучшения управления документами в ЦЦК.

Цель работы: разработать и внедрить информационную систему отслеживания статуса документов, которая позволит улучшить управление документами в Центре цифровых компетенций (ЦЦК), обеспечив их эффективное хранение, доступность и отслеживание.

Для достижения необходимой цели нужно выполнить такие задачи как:

– анализ текущей ситуации: Исследование и анализ существующих

методов управления документами в ЦЦК, выявление их недостатков и проблем;

- исследование и анализ требований: определение основных функциональных требований к системе, анализ потребностей пользователей и организации, исследование существующих решений и технологий;
- разработка архитектуры системы: определение структуры базы данных для хранения документов, разработка архитектуры пользовательского интерфейса, определение модулей системы и их взаимодействия;
- разработка пользовательского интерфейса: реализация функционала для просмотра, добавления, редактирования и удаления документов, разработка функционала поиска и фильтрации документов;
- тестирование и оптимизация системы: проведение функционального и юзабилити тестирования для проверки корректности работы всех модулей системы, тестирование производительности системы под нагрузкой, оптимизация системы для улучшения скорости работы и эффективности использования ресурсов.

Бакалаврская работа состоит из введения, трех глав и заключения.

В первой главе мы рассмотрим фундаментальные принципы и концепции, лежащие в основе информационных систем, а также определим ключевые требования и цели, которые должна удовлетворять наша система.

Во второй главе описывается процесс проектирования системы, включая выбор архитектуры, определение требований к функциональности и интерфейсам, а также разработку схемы базы данных.

В третьей главе описывается процесс реализации и тестирование ИС.

В заключении сделаны основные выводы и итоги по проделанной работе.

Глава 1 Теоретические основы ИС и анализ требований

1.1 Общие сведения о Тольяттинском государственном университете

29 мая 2001 года, в результате решения правительства Российской Федерации, был учрежден Тольяттинский государственный университет (ТГУ). Этот университет был основан на базе Тольяттинского политехнического института и Тольяттинского филиала Самарского государственного педагогического университета.

На сегодняшний день ТГУ является одним из ведущим техническим вузом России, занимающим ключевые позиции среди технических учреждений образования, предлагая высококачественное образование в различных направлениях. ТГУ имеет статус опорного вуза в Самарской области и входит в Ассоциацию классических университетов России, что подчеркивает его значимость в образовательной сфере.

Основная цель ТГУ – подготовка практиков, которые готовы решать профессиональные задачи сразу после выпуска без дополнительного обучения, стажировок. Все студенты имеют портфолио с академическими достижениями, опытом в исследованиях и участии в работающих проектах по специальности.

ТГУ состоит из 10 институтов и 47 кафедр, предлагающих обучение по различным направлениям. В университете работает более 2000 преподавателей, больше половины из которых имеют послевузовские научные степени. В ТГУ обучается больше 10 тысяч студентов.

Структура университета, представленная на рисунке 1.

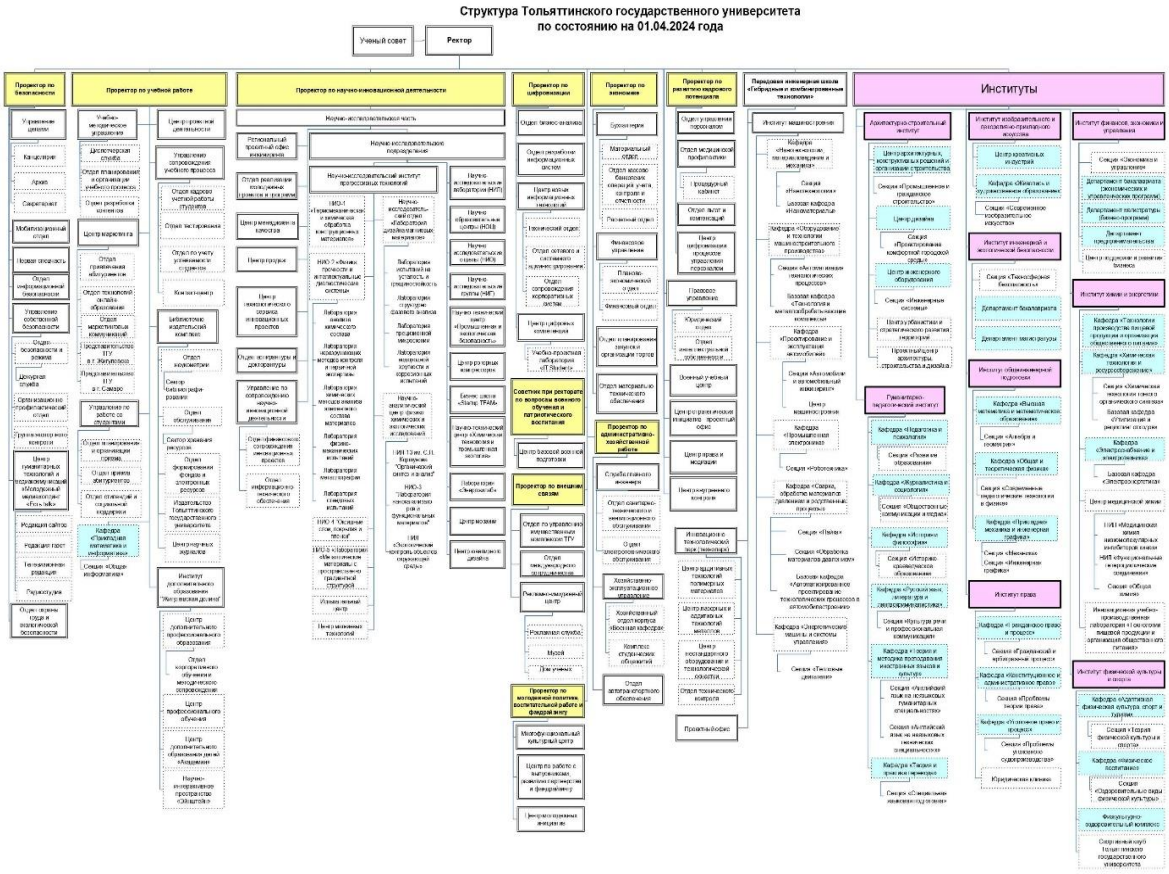


Рисунок 1 – Структура Тольяттинского государственного университета

Структура демонстрирует иерархию управления, где ученый совет и ректор руководят проректорами по направлениям, а также управляющими структурными подразделениями, институтами, и службами.

1.2 Общие сведения о ЦЦК

Центр цифровых компетенций (ЦЦК) в Тольяттинском государственном университете (ТГУ) является важным элементом в структуре университета, направленным на развитие и укрепление цифровых навыков и компетенций студентов, аспирантов и сотрудников университета. Основанный с целью поддержки и продвижения цифровой трансформации в образовательном процессе и научных исследованиях, ЦЦК занимается широким спектром

деятельности, направленной на интеграцию цифровых технологий в образовательную и научную деятельность университета.

ЦЦК в ТГУ занимается созданием англоязычной образовательной программы, что является сложным и многогранным процессом. Это включает в себя разработку контента, найм преподавателей и установление партнерских программ с IT-компаниями. Эти задачи требуют глубокого понимания образовательных процессов, современных технологий и международного образовательного пространства.

Разработка контента для англоязычной образовательной программы начинается с определения образовательных целей и методов обучения. Это включает в себя анализ текущих образовательных программ, исследование лучших практик в области образования, а также разработку новых методик и подходов к обучению. ЦЦК работает над созданием учебных материалов, методических рекомендаций и интерактивных образовательных ресурсов, которые могут быть адаптированы для использования в англоязычной образовательной программе.

Найм преподавателей является еще одной важной задачей ЦЦК. Это включает в себя поиск и отбор кандидатов, проведение собеседований, а также организацию процесса трудоустройства. ЦЦК должен обеспечить высокое качество образования, выбирая преподавателей, которые обладают необходимыми навыками и опытом в области образования. Это также включает в себя разработку и реализацию программ подготовки преподавателей, которые могут помочь им адаптироваться к новым методам обучения и современным образовательным технологиям.

Установление партнерских программ с IT-компаниями также является ключевым аспектом работы ЦЦК. Это включает в себя поиск и выбор партнеров, которые могут предложить современные технологии и инструменты для образовательного процесса, а также организацию совместных проектов и мероприятий. ЦЦК работает над созданием условий для эффективного сотрудничества с IT-компаниями, что позволяет

интегрировать новые технологии в образовательный процесс и обеспечивать актуальность образовательных программ.

Дополнительные направления деятельности ЦЦК в ТГУ включают:

- организация образовательных программ и курсов по цифровым технологиям и компетенциям, направленных на подготовку специалистов, способных эффективно использовать цифровые инструменты и платформы в различных сферах деятельности;
- проведение научно-исследовательских работ в области цифровых технологий, включая разработку новых методов и подходов к использованию цифровых ресурсов в образовании и науке;
- разработка и внедрение инновационных образовательных и научных проектов, направленных на применение цифровых технологий для улучшения качества образования и научных исследований;
- создание и поддержка цифровой инфраструктуры университета, включая разработку и внедрение цифровых образовательных платформ, систем управления базами данных и других цифровых ресурсов;
- организация мероприятий и семинаров по актуальным вопросам цифровой трансформации, направленных на повышение квалификации преподавателей и студентов, а также на обмен опытом и знаниями в области цифровых технологий;
- сотрудничество с отраслевыми партнерами для реализации совместных образовательных и научных проектов, направленных на развитие цифровой экономики и цифровой культуры;
- поддержка и развитие научно-образовательного сообщества университета в области цифровых компетенций, включая организацию конкурсов, стажировок и грантов для студентов и аспирантов.

Таким образом, ЦЦК в ТГУ выполняет множество важных функций, направленных на улучшение качества образования и подготовку студентов к

современным вызовам. Для достижения этих целей ЦЦК активно использует современные технологии и инструменты. Это включает в себя использование систем управления проектами для координации работы над проектами, использование платформ для дистанционного обучения для обеспечения доступа к образовательным материалам и интерактивным образовательным ресурсам, а также использование аналитических инструментов для мониторинга и оценки эффективности образовательных программ. На рисунке 2 представлена обобщенная структура ТГУ.

Рисунок 2 – Обобщённая структура Тольяттинского государственного университета

ЦЦК в ТГУ играет важную роль в развитии образовательной деятельности университета и подготовке студентов к современным вызовам. Это включает в себя создание англоязычной образовательной программы, найм преподавателей, установление партнерских программ с IT-компаниями, а также развитие образовательной деятельности университета в целом. ЦЦК работает над созданием условий для успешного развития образовательной

деятельности и подготовки квалифицированных специалистов, что является ключевым для успеха университета в современном образовательном пространстве.

1.3 Классификация и принцип работы информационных систем

В контексте информационных систем, под системой понимают сложную структуру, объединяющую различные элементы с целью достижения определенных задач. Эти системы могут существенно отличаться по своему составу и основным целям. В контексте деятельности организации, ИС рассматривается как программное обеспечение, реализующее её деловую стратегию и бизнес-процессы. Желательной целью является создание и развертывание единой корпоративной информационной системы, удовлетворяющей информационные потребности всех сотрудников, служб и подразделений организации.

Для описания и применения всех этих средств автоматизации сбора, хранения, поиска и представления информации используется понятие «информационная система». ИС представляет собой сложную и многогранную структуру, которая включает в себя различные компоненты и подсистемы, работающие вместе для достижения поставленных целей [11].

Информационные системы являются динамичными и развивающимися структурами, которые могут быть анализированы, построены и управляемы на основе общих принципов построения систем. При их создании ключевым является использование системного подхода, который обеспечивает комплексное и целостное видение процесса обработки информации. Выходной продукцией таких систем является информация, которая служит основой для принятия решений. Важно понимать информационную систему как часть человеко-компьютерной системы обработки информации, где взаимодействие между людьми и технологиями играет центральную роль [17].

Процессы, обеспечивающие работу информационной системы любого

назначения, условно можно представить в виде схемы, состоящей из блоков [10]:

- ввод информации из внешних или внутренних источников,
- обработка входной информации и представление ее в удобном виде,
- вывод информации для представления потребителям или передачи в другую систему,
- обратная связь – это информация, переработанная людьми данной организации для коррекции входной информации.

На рисунке 3 показаны процессы, обеспечивающие работу информационной системы.

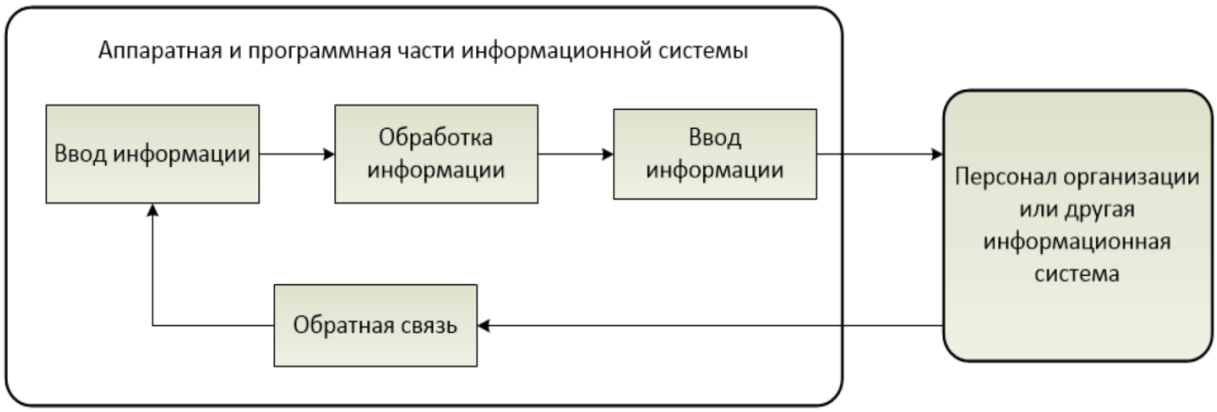


Рисунок 3 - Процессы в информационной системе

Информационные системы представляют собой сложные и многогранные структуры, которые играют ключевую роль в современном мире. Они обеспечивают эффективное управление информацией и ресурсами, улучшают принятие решений и повышают эффективность работы. Для достижения максимальной эффективности и адаптивности, информационные системы должны быть разработаны с учетом современных подходов к разработке программного обеспечения. Само определение информационной системы и ее классификация остаются предметом дискуссий и исследований, что подчеркивает ее важность и роль в современном мире [21].

Информационные системы представляют собой сложные структуры, включающие в себя различные компоненты, такие как аппаратное обеспечение, программное обеспечение, данные, процессы и пользователи, которые работают вместе для достижения поставленных целей. В зависимости от различных критериев информационные системы могут быть классифицированы по различным параметрам, включая функциональность, масштаб, принцип деятельности и характер информации, с которой работает система [21].

Одним из ключевых принципов классификации информационных систем является классификация по характеру обработки данных. В этом контексте информационные системы делятся на информационно-справочные или информационно-поисковые ИС, в которых нет сложных алгоритмов обработки данных, а целью системы является поиск и выдача информации в удобном виде. К таким системам относятся, например, системы поиска в интернете или базы данных для поиска информации. С другой стороны, существуют информационные системы обработки данных или решающие ИС, в которых данные подвергаются обработке по сложным алгоритмам. К таким системам в первую очередь относятся автоматизированные системы управления и системы поддержки принятия решений, которые используются для анализа данных и принятия обоснованных решений на основе анализа.

Классификация информационных систем по степени автоматизации также играет важную роль в их понимании. В этом контексте информационные системы делятся на автоматизированные, в которых автоматизация может быть неполной, то есть требуется постоянное вмешательство персонала, и автоматические, в которых автоматизация является полной, то есть вмешательство персонала не требуется или требуется только эпизодически. Это позволяет лучше понять, насколько информационная система зависит от человеческого вмешательства и насколько она может работать автономно [23].

Одним из ключевых принципов классификации ИС является

классификация по функциональности, которая позволяет разделить их на операционные, административные, управленческие и другие. Это позволяет лучше понять, какие задачи решает каждая система и как она вписывается в общую структуру организации. Кроме того, существует классификация по масштабу, которая позволяет определить, насколько широко используется система внутри организации, и классификация по принципу деятельности, которая указывает на специфику деятельности, для которой предназначена система.

Существует также классификация информационных систем по сфере применения, которая учитывает широкий спектр областей, в которых информационные системы могут быть использованы. Это включает в себя бухгалтерскую отчетность, поисковики, разработки в области искусственного интеллекта, исследования в космосе и многие другие сферы деятельности. Такая классификация позволяет лучше понять, как информационные системы могут быть адаптированы и использованы в различных контекстах, чтобы решать актуальные задачи и повышать эффективность работы [17].

Корпоративные информационные системы (КИС) представляют собой один из наиболее распространенных типов информационных систем, используемых в организациях различных размеров и отраслей. Они могут быть классифицированы по целевой задаче, по рынку сбыта и по сложности структуры. КИС могут быть слабо интегрированными, что позволяет разделить функции системы на автономные сервисы, ориентированные на многообразие различных типов данных. Это обеспечивает широкие возможности самостоятельной работы отдельных пользователей, не связанных обязательными алгоритмами действий с данными. КИС с сильно интегрированной архитектурой представляют собой набор приложений, но отличаются единством интерфейса, единством форматов представления данных, и жесткой связью между отдельными приложениями. Взаимосвязь между приложениями должна точно соответствовать бизнес-процессам, которые просто «прописываются» в структуре КИС, что значительно

облегчает работу с ними неподготовленным пользователям. Данные в КИС такого типа практически не дублируются, и могут быть представлены во всем многообразии своих взаимосвязей, что исключительно важно при осуществлении аналитической деятельности и сквозном управленческом контроле [18][20].

Классификация информационных систем позволяет лучше понять их структуру, функциональность и область применения, что является ключевым для эффективного управления информацией и ресурсами в организации. Различные типы информационных систем, такие как корпоративные, производственные, финансовые, образовательные и другие, играют важную роль в современном мире, обеспечивая эффективное управление ресурсами и информацией, улучшая принятие решений и повышая эффективность работы.

Принципы работы информационных систем являются основой для их эффективного функционирования и достижения целей. Эти принципы определяют, как система собирает, обрабатывает, хранит и предоставляет информацию, а также как она взаимодействует с пользователями и другими системами. В зависимости от конкретных задач и целей, информационные системы могут применять различные подходы и методы, но все они основываются на нескольких ключевых принципах.

Одним из основных принципов работы информационных систем является автоматизация процессов. Это означает, что большинство задач, связанных с обработкой информации, выполняются автоматически с помощью программного обеспечения, минимизируя необходимость ручного вмешательства и увеличивая скорость и точность обработки данных. Автоматизация позволяет сократить время на выполнение рутинных задач, освободить ресурсы персонала для более важных задач и улучшить качество работы, избегая ошибок, которые могут возникнуть при ручном вводе данных.

Другим важным принципом является централизация данных. В централизованной информационной системе все данные хранятся и обрабатываются в одном месте, что облегчает их управление, обеспечивает

согласованность и упрощает доступ к информации для всех пользователей. Централизация также упрощает обмен информацией между различными подразделениями организации и с внешними партнерами, что важно для эффективного управления и координации деятельности.

Принцип модульности также играет ключевую роль в работе информационных систем. Модульная архитектура позволяет разрабатывать и внедрять отдельные компоненты системы независимо друг от друга, что упрощает процесс обновления и модернизации системы. Модульность также обеспечивает гибкость системы, позволяя легко адаптировать ее под изменяющиеся требования и условия работы [25].

Совместно с этим, принцип безопасности является неотъемлемой частью работы информационных систем. Системы должны обеспечивать защиту данных от несанкционированного доступа, изменения или уничтожения, а также защищать конфиденциальность и целостность информации. Безопасность может включать в себя различные механизмы, такие как шифрование данных, использование аутентификации и авторизации, а также регулярное обновление и патчинг программного обеспечения.

Наконец, принцип масштабируемости является важным для информационных систем, особенно в современном мире, где требования к обработке данных и объему информации постоянно растут. Масштабируемость означает способность системы адаптироваться к увеличению объема данных и нагрузки, обеспечивая стабильную и эффективную работу даже при значительном увеличении объемов обрабатываемой информации.

Принципы работы информационных систем, такие как автоматизация, централизация, модульность, безопасность и масштабируемость, являются фундаментом для их эффективного функционирования. Они обеспечивают, что информационные системы могут эффективно поддерживать деятельность организации, улучшать принятие решений и повышать эффективность работы.

1.4 Анализ требований к системе отслеживания статуса документов ЦЦК

Центр цифровых компетенций в ТГУ играет ключевую роль в развитии цифровой образовательной программы, ориентированной на создание англоязычных образовательных материалов. Это включает в себя разработку контента, найм преподавателей, а также установление партнерских программ с IT-компаниями. Однако, в рамках своей деятельности ЦЦК сталкивается с рядом проблем, связанных с отслеживанием статуса договоров и актов.

На рисунке 4 представлена высокоуровневый процесс согласования договора.

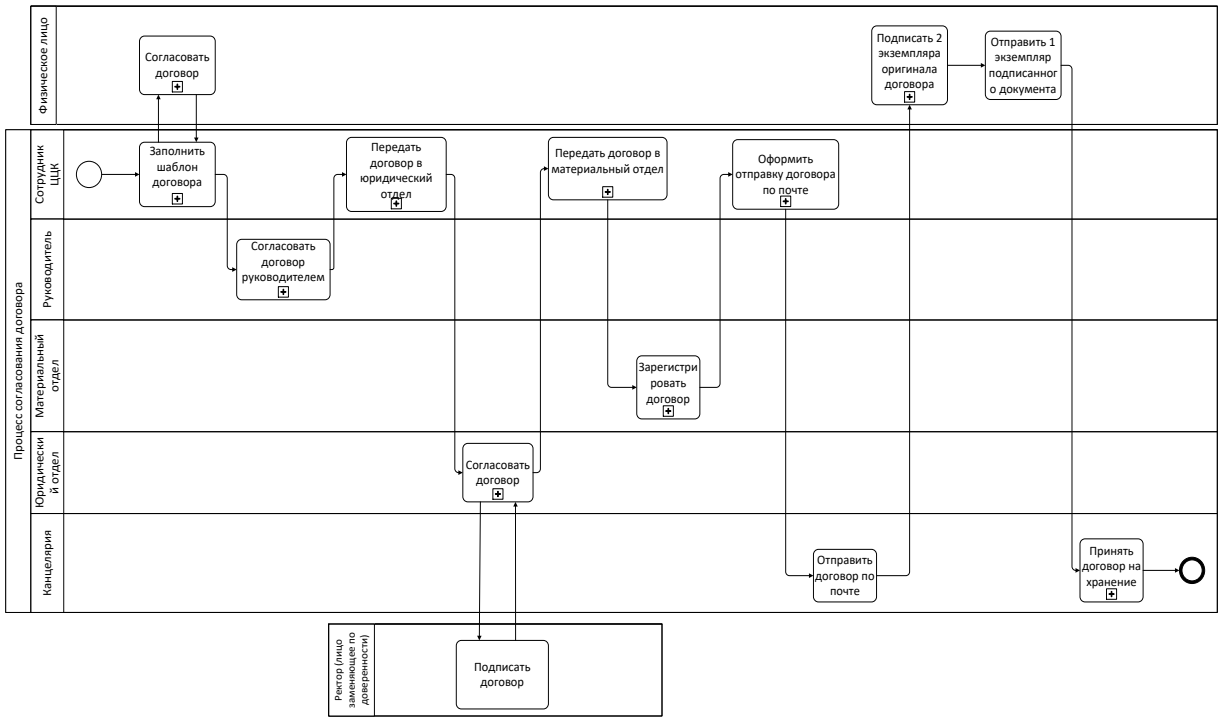


Рисунок 4 – Высокоуровневый процесс согласования договора в ЦЦК

Процесс подписания договоров и актов начинается с электронного обмена скан-копиями документов. Этот метод позволяет обеспечить удобство и скорость процесса, однако он также создает проблемы с отслеживанием статуса документов. После того как документы подписаны, они пересылаются

оригиналами по почте. Этот процесс требует значительного времени и ресурсов, а также может привести к ошибкам при передаче документов.

После подписания оригиналов документов ЦЦК передает их во внутренние службы ТГУ руками. Этот метод передачи документов является устаревшим и неэффективным, особенно в условиях, когда требуется быстрое и точное отслеживание статуса документов. В современном мире, где цифровые технологии играют ключевую роль в управлении документами и информацией, такой подход может привести к задержкам и ошибкам.

Для решения этих проблем ЦЦК может рассмотреть возможность внедрения цифровых решений для управления документами и информацией. Это может включать в себя использование систем управления документами, которые обеспечивают возможность отслеживания статуса документов в реальном времени. Такие системы могут значительно упростить процесс управления документами и улучшить эффективность работы ЦЦК.

Требования к системе:

- цель системы: Улучшение процесса управления документами в ЦЦК, обеспечивая точное отслеживание статуса договоров и актов, а также предотвращение потери данных документов.
- основные функции: управление циклом жизни документов, включая хранение, отслеживание статуса и передачу оригиналов; обеспечение доступности информации о статусе документов для всех заинтересованных сторон.
- требования к пользовательскому интерфейсу: интуитивно понятный интерфейс для удобства использования; возможность просмотра текущего статуса документов в реальном времени; инструменты для управления доступом к информации о документах.
- требования к безопасности: защита конфиденциальности и целостности данных документов; возможность аутентификации и авторизации пользователей.
- требования к масштабируемости и производительности: способность

обрабатывать большое количество документов без снижения производительности; обеспечение надежности и доступности.

– требования к поддержке и обслуживанию: предоставление документации и обучающих материалов для пользователей.

Разработка информационной системы с учетом вышеуказанных требований позволит значительно улучшить процесс управления документами в ЦЦК. Эта система будет не только обеспечивать точное отслеживание статуса договоров и актов, но и предотвращать потерю данных документов, что критически важно для эффективного функционирования организации Создание такой ИС станет ключевым шагом в модернизации процессов управления документами в ЦЦК, обеспечивая повышение эффективности работы и сокращение рисков связанных с обработкой документов.

Вывод по первой главе

В первой главе были рассмотрены теоретические основы разработки информационных систем. Было проведено детальное исследование требований к информационной системе отслеживания документов ЦЦК, что позволило определить ключевые функциональные и нефункциональные требования, а также выявить основные проблемы и ограничения текущей системы. Анализ требований показал необходимость внедрения новых технологий и методик для повышения эффективности работы с документами, а также улучшения взаимодействия между различными участниками процесса.

Глава 2 Проектирование, технологические решения и выборы для информационной системы отслеживания документов ЦЦК

2.1 Выбор архитектурного подхода

Архитектура информационной системы отслеживания статуса документов для ЦЦК представляет собой комплексное решение, обеспечивающее эффективное управление документами и их статусами. В данном разделе мы подробно рассмотрим архитектурный подход, выбранный для проекта, а также ключевые компоненты системы, их взаимодействие и функциональность.

При разработке информационной системы отслеживания статуса документов для ЦЦК было принято решение о выборе монолитной архитектуры. Монолитная архитектура представлена на рисунке 5.

Этот выбор обусловлен рядом причин:

- монолитная архитектура позволяет обеспечить высокую степень интеграции компонентов системы, что важно для системы отслеживания статуса документов, где все компоненты должны тесно взаимодействовать друг с другом [19];
- централизованное управление: В монолитной архитектуре все компоненты системы интегрированы и управляются в рамках одного приложения. Это обеспечивает централизованное управление данными и функциональностью, что упрощает разработку и поддержку системы, так как могут быть легко модифицированы [16];
- высокая производительность. Монолитная архитектура позволяет обеспечить высокую производительность системы, так как все компоненты системы работают в рамках одной программной среды;
- общие ресурсы. В монолитной архитектуре все компоненты системы используют общие ресурсы, такие как базы данных и кэши. Это позволяет оптимизировать использование ресурсов и улучшить

производительность системы [7];

Хотя монолитная архитектура может представлять собой преимущество в некоторых случаях, важно учитывать ее ограничения в контексте масштабируемости и производительности:

– ограничения масштабируемости: монолитные приложения могут столкнуться с проблемами при масштабировании, особенно при увеличении нагрузки. Это может потребовать дополнительных ресурсов и оптимизаций для обеспечения производительности [11];
– производительность: в монолитной архитектуре все компоненты системы работают в рамках одного процесса, что может привести к увеличению времени ответа при обработке запросов. Однако, с правильной оптимизацией и использованием кэширования, производительность может быть достаточно высокой для большинства приложений.

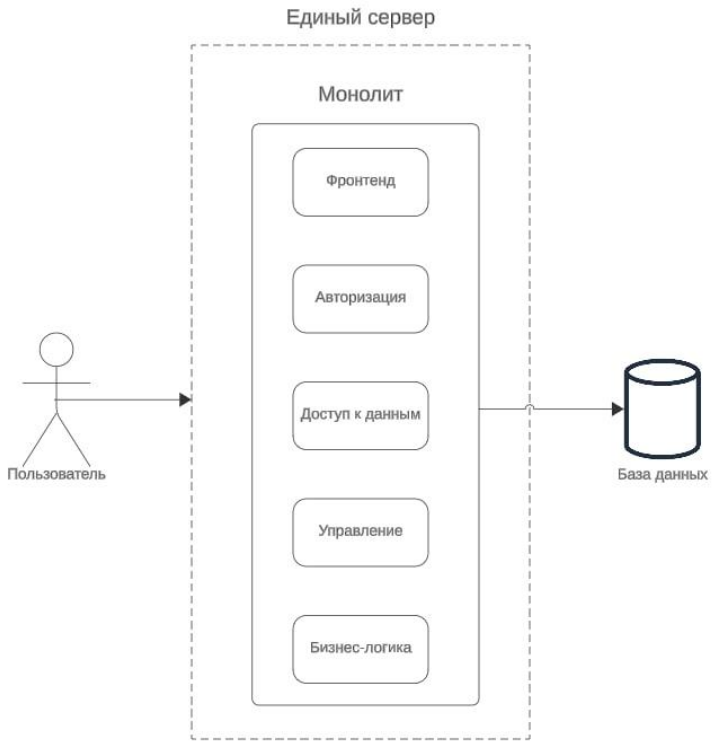


Рисунок 5 — Монолитная архитектура

В монолитной архитектуре компоненты системы взаимодействуют друг с другом через определенные интерфейсы и API. Это обеспечивает четкую границу между компонентами и позволяет изменять или обновлять отдельные компоненты без влияния на остальную систему. Для взаимодействия компонентов в монолитной архитектуре также используются ORM, максины и наследование. ORM позволяет легко взаимодействовать с базой данных, обеспечивая абстракцию от SQL и упрощая разработку. Это особенно полезно в монолитной архитектуре, где все компоненты системы работают в рамках одного приложения. Использование миксинов и наследования для расширения функциональности моделей и представлений, что позволяет легко интегрировать новые компоненты и функции в систему [5].

Основные компоненты, которые будут использоваться в нашим проекте для отслеживания статуса документов включают в себя:

- фронтенд: этот компонент обеспечивает пользовательский интерфейс системы, позволяя пользователям взаимодействовать с системой через веб-браузер;
- бэкенд: бэкенд представляет собой серверную часть системы, которая обрабатывает запросы от фронтенда, взаимодействует с базой данных и выполняет основную логику обработки данных. Бэкенд разработан на языке программирования Python с использованием фреймворка Django, что обеспечивает высокую производительность и масштабируемость системы [6];
- база данных: база данных используется для хранения и управления данными о документах и их статусах. Выбор СУБД осуществлялся с учетом требований к производительности, надежности и масштабируемости. В качестве СУБД выбрана PostgreSQL, что позволяет обеспечить высокую производительность и надежность хранения данных [24].

Система отслеживания статуса документов для ЦЦК обеспечивает следующие функции:

– регистрация новых документов и их статусов,
– обновление статусов документов в реальном времени,
– поиск документов по различным критериям,
– генерация отчетов о статусах документов,
– управление доступом пользователей к документам.

Взаимодействие между компонентами системы будет осуществляться через REST API, разработанное на основе стандартов HTTP. Это позволяет обеспечить гибкое и масштабируемое взаимодействие между фронтендом и бэкендом, а также между бэкендом и базой данных [12].

2.2 Выбор Django как фреймворка для разработки

В качестве инструмента для реализации информационной системы отслеживания статуса документов для ЦЦК выбран фреймворк Django. Django, как и любой другой фреймворк, имеет свои преимущества и недостатки, которые следует учитывать при выборе подходящего инструмента для разработки. Выбор Django для нашей разработки обусловлен рядом преимуществ, которые делают его идеальным выбором для такого проекта:

– быстрая разработка: Django позволяет разработчикам быстро создавать и разрабатывать веб-приложения, благодаря встроенным инструментам и модульному подходу. Также имеется много дополнительный готовых пакетов, которых можно подключать и использовать в процессе разработки. Это сокращает время разработки и позволяет сосредоточиться на бизнес-логике [3];
– мощная архитектура: Django использует архитектуру Model-View-Template (MVT), что обеспечивает четкое разделение между бизнес-логикой, представлением и контроллером. Это упрощает разработку, поддержку и расширение приложения [3][9];
– безопасность: Django предоставляет встроенные механизмы защиты от распространенных угроз, таких как SQL-инъекции и кросс-

сайтовый скриптинг. Это обеспечивает безопасность данных и пользователей;

- масштабируемость: Django легко масштабируется, что позволяет легко добавлять новые функции и модули без значительных изменений в существующем коде. Это делает его идеальным выбором для проектов, которые могут расти и развиваться [22];
- развитая экосистема: Django обладает богатой экосистемой, включающей множество библиотек и плагинов, что упрощает реализацию различных функций и ускоряет процесс разработки [23];
- административная панель: Django автоматически генерирует административную панель для управления данными, что существенно упрощает процесс администрирования и управления контентом.

Однако, стоит учесть и некоторые недостатки Django:

- Django может быть избыточным для маленьких проектов, где его использование может привести к излишней сложности и накладным расходам. Это связано с тем, что Django оптимизирован для быстрой разработки крупномасштабных приложений, что требует значительного времени серверной обработки и пропускной способности [4][8],
- Django может быть сложным для настройки, из-за его строгой структуры и предпочтений в отношении определенных подходов к разработке,
- несмотря на его модульность, Django все же имеет монолитную архитектуру, что может ограничивать гибкость в некоторых случаях,
- в отличие от некоторых других фреймворков, Django не использует конвенций, что может вызвать замедление прогресса и отторжение со стороны некоторых программистов.

Django также отличается от других фреймворков своей архитектурой, что делает процесс передачи данных по Интернету более простым и быстрым.

Это обеспечивает быстрое обработка данных и поддерживает высокую скорость работы сервера, что является ключевым преимуществом для веб-разработки [3][8].

Стоит отметить еще тот факт, что Django обладает отличной документацией, что делает его популярным выбором среди программистов. Также Django имеет большое сообщество разработчиков, которые имеют глубокие знания Django. Наличие большого сообщества имеет свои преимущества, так как это упрощает поиск ответов на проблемы.

2.3 Выбор СУБД для хранения данных

В контексте проектирования информационной системы выбор системы управления базами данных (СУБД) является критически важным шагом. СУБД определяет, как данные будут храниться, обрабатываться и извлекаться, что напрямую влияет на производительность, масштабируемость и безопасность системы. В данном случае, для проекта выбрана PostgreSQL, одна из самых мощных и гибких СУБД, известная своей надежностью, поддержкой распределенных систем и продвинутыми функциями безопасности [1].

PostgreSQL предлагает широкий спектр возможностей для проектирования схемы базы данных, включая поддержку различных типов данных, таких как числовые, строковые, даты и временные метки, а также сложные типы данных, такие как массивы и JSON. Это позволяет разработчикам создавать сложные структуры данных, которые могут эффективно хранить и обрабатывать информацию, необходимую для информационной системы [1][2].

Безопасность данных является неотъемлемой частью любой информационной системы. PostgreSQL предоставляет ряд встроенных механизмов для обеспечения безопасности данных, включая шифрование данных, управление доступом и аудит. Рассмотрим их подробнее:

– PostgreSQL поддерживает шифрование данных на уровне диска и в базе данных, что позволяет защитить конфиденциальную информацию от несанкционированного доступа [15].
– PostgreSQL позволяет управлять доступом к базе данных и отдельным объектам базы данных, используя роли и привилегии. Это обеспечивает гибкость в управлении правами доступа к данным [15].
– PostgreSQL предоставляет возможности для аудита действий в базе данных, включая отслеживание изменений в данных и выполнения операций. Это помогает обеспечить прозрачность и ответственность за действия, совершаемые в системе [14].

Необходимо еще указать тот факт, что фреймворк Django также предоставляет ряд встроенных механизмов для обеспечения безопасности данных, включая защиту от SQL-инъекций, управление пользователями и группами, аутентификацию и авторизацию, а также защиту от XSS и CSRF атак:

– Django ORM обеспечивает защиту от SQL-инъекций, используя параметризованные запросы и автоматическую экранировку входных данных. Это предотвращает возможность выполнения вредоносного SQL-кода, который может привести к несанкционированному доступу к данным [13];
– Django предлагает мощную систему управления пользователями и группами, позволяющую легко реализовывать различные уровни доступа и управление правами доступа к данным;
– Django предоставляет встроенные механизмы для аутентификации пользователей и управления их правами доступа к различным ресурсам системы. Это включает в себя хеширование паролей, управление сессиями и поддержку двухфакторной аутентификации;
– Django включает в себя механизмы для защиты от XSS и CSRF атак, используя Content Security Policy (CSP) и другие техники безопасности.

В дополнение к встроенным механизмам безопасности PostgreSQL, также можно использовать различные практики и инструменты для обеспечения безопасности данных, такие как использование безопасных соединений (SSL/TLS), регулярное обновление и патчинг базы данных, а также применение принципов безопасного программирования при разработке приложений, работающих с базой данных [3][5].

Реализация этих механизмов безопасности является важным шагом в обеспечении защиты информационной системы отслеживания статуса документов для ЦЦК.

Выбор PostgreSQL как СУБД и разработка соответствующей структуры данных для отслеживания статуса документов, а также реализация механизмов безопасности данных, являются ключевыми шагами в проектировании информационной системы. Эти шаги обеспечивают надежность, производительность и безопасность системы, что является критически важным для успешной реализации проекта.

2.4 Определение структуры базы данных для отслеживания статуса документов ЦЦК

Определение структуры базы данных для отслеживания статуса документов в информационной системе для ЦЦК является ключевым этапом в разработке системы и ее успешной работы. Схема должна быть гибкой и масштабируемой, обеспечивая легкое добавление новых сущностей и атрибутов. Она должна также обеспечивать целостность данных, предотвращая дублирование и несоответствия данных в разных местах. Ключевые аспекты проектирования схемы базы данных включают определение типов данных, создание отношений между сущностями и применение ограничений для обеспечения целостности данных.

Структура базы данных для нашего проекта показана на рисунке 6:

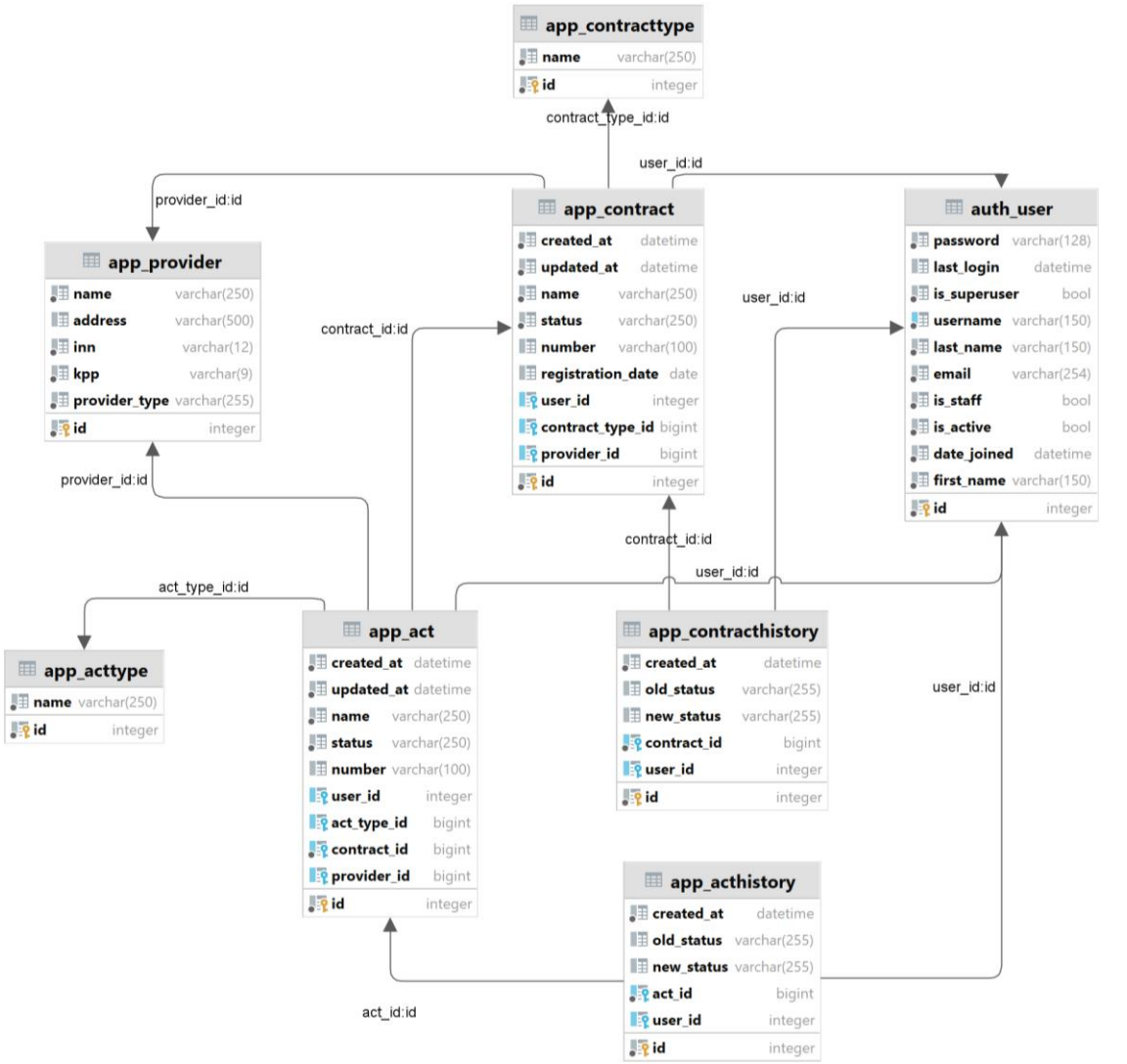


Рисунок 6 — Схема базы данных сервиса отслеживания статуса документов ЦЦК

Техническая реализация структуры базы данных приведена в Приложении А.

Эта структура предполагает, что каждый документ имеет тип, может быть связан с одним или несколькими актами, имеет поставщика, историю изменений статусов и связан с пользователем, который является сотрудником ЦЦК, и имеет определенную роль в процессе отслеживания.

Договор является центральной сущностью в системе, поскольку он представляет собой основу для отслеживания всех документов и их статусов.

Договор должен содержать информацию о дате подписания, дата обновления, статус, номер и дата регистрации. Также важно учитывать возможность связывания договора с конкретным типом договора, поставщиком, истории, пользователем и может быть связан с одним или несколькими актами.

Тип договора позволяет классифицировать договоры по различным критериям, таким как вид услуг, товаров или работ, которые предоставляются. Это обеспечивает возможность быстрого поиска и анализа документов по определенным категориям. Тип договора может быть связан с несколькими договорами и содержать дополнительные атрибуты, такие как описание услуг или товаров.

Акт является документом, подтверждающим выполнение условий договора. Он может содержать информацию о выполненных работах, предоставленных товарах или услугах, а также о дате и месте выполнения. Акт должен быть связан с конкретным договором, что позволяет системе отслеживать выполнение условий договора и обеспечивать его соблюдение. Также возможно акт будет содержать ссылки на дополнительные документы или материалы. Акты также будут связаны с пользователями, которые их создали или подтвердили, что позволяет системе отслеживать историю изменений статусов документов.

Поставщик представляет собой организацию или индивидуума, предоставляющего товары или услуги в соответствии с договором. Информация о поставщике включает в себя название, адрес и контактные данные. Поставщик может быть связан с несколькими договорами и актами. Поставщиков в системе могут добавлять или обновлять пользователи с соответствующими правами. Поставщики также могут быть классифицированы по различным критериям, что позволяет системе управлять отношениями с поставщиками на разных уровнях.

Сущность "История" отслеживает изменения статусов документов и актов, и их жизненный цикл. Она может содержать информацию о дате изменения статуса, о пользователе, совершившем изменения и причинах

изменения. История позволяет восстанавливать предыдущие состояния документов и анализировать их изменения. История также может быть связана с пользователями, что позволяет системе отслеживать их вклад в изменение статусов документов.

Пользователь (сотрудник ЦЦК) представляет собой сущность, отвечающую за взаимодействие с системой. Каждый пользователь должен иметь уникальный идентификатор, имя, контактные данные и роль в системе. Роль определяет права доступа пользователя к различным функциям системы. Пользователи также могут быть связаны с договорами и актами, что позволяет системе отслеживать их вклад в управление документами и отслеживать историю их действий.

Роль определяет права доступа пользователя в системе. Она может включать в себя различные уровни доступа, такие как просмотр, редактирование, удаление документов, управление пользователями и т.д. Роль может быть назначена одному или нескольким пользователям и может быть изменена в зависимости от изменения требований к системе.

Связи между сущностями обеспечивают гибкость и масштабируемость системы, позволяя эффективно управлять информацией о договорах, актах, пользователях, ролях и поставщиках. Они также обеспечивают прозрачность процессов и возможность отслеживания изменений, что важно для аудита и отчетности. Связи между сущностями будут следующими**:**

- Договор и Акт: договор и акт взаимосвязаны таким образом, что договор может быть связан с одним или несколькими актами, подтверждающими выполнение условий договора. Это позволяет системе отслеживать выполнение договорных обязательств и обеспечивает прозрачность процесса. Акты могут быть связаны с несколькими договорами, что указывает на их многостороннюю природу и способствует анализу и управлению договорами.
- Договор и Поставщик: связь между договором и поставщиком обеспечивает управление отношениями с поставщиками, позволяя

системе отслеживать, какие поставщики участвуют в договорах и какие условия установлены. Это важно для обеспечения качества услуг и товаров, а также для контроля над поставками.

- Пользователь и Договор: пользователи могут быть связаны с договорами, что позволяет системе отслеживать их вклад в управление договорами. Это может включать в себя создание, редактирование, подтверждение и отклонение договоров. Связь пользователя с договором также позволяет системе отслеживать историю изменений договора, что важно для аудита и отчетности;
- Пользователь и Акт: пользователи могут быть связаны с актами, что позволяет системе отслеживать их вклад в управление актами. Это может включать в себя создание, редактирование, подтверждение и отклонение актов. Связь пользователя с актом также позволяет системе отслеживать историю изменений акта, что важно для аудита и отчетности;
- История и Пользователь: история изменений статусов документов может быть связана с пользователями, что позволяет системе отслеживать их вклад в изменение статусов документов. Это важно для аудита и отчетности, позволяя определить, кто и когда вносил изменения в документы;
- Роль и Пользователь: роль определяет права доступа пользователя в системе, что позволяет системе управлять доступом пользователей к различным функциям системы. Это важно для обеспечения безопасности и конфиденциальности информации, а также для управления доступом к различным ресурсам и функциям системы;
- Тип Договора и Договор: тип договора может быть связан с несколькими договорами, что позволяет системе классифицировать и анализировать договоры по различным критериям. Это важно для управления и анализа договоров, позволяя системе отслеживать и анализировать различные типы договоров и их влияние на

организацию;

- История и Договор: история может записывать каждое изменение статуса договора, включая дату изменения, пользователя, совершившего изменение, и причину изменения. Это позволяет увидеть предыдущие состояния договоров и анализировать их изменения;
- История и Акт: аналогично истории договоров, история может отслеживать изменения статусов актов, включая дату изменения, пользователя, совершившего изменение, и причину изменения. Это позволяет увидеть предыдущие состояния актов и анализировать их изменения.

Данная структура моделей позволяет создать гибкую и масштабируемую базу данных для сервиса отслеживания статуса документов и обеспечивают основу для хранения и управления информацией о документах, их типах, актах, поставщиках, истории изменений статусов, пользователях и их ролях в процессе отслеживания.

Вывод по второй главе

Во второй главе были разработаны архитектурные решения и выбраны технологии для создания информационной системы отслеживания статуса документов ЦЦК. Были определены основные компоненты системы, включая базу данных, серверную часть, клиентскую часть и механизмы взаимодействия между ними. В результате были выбраны фреймворк Django и база данных PostgreSQL как технологических решений для разработки информационной системы отслеживания статуса документов ЦЦК.

Глава 3 Разработка и тестирование информационной системы отслеживания статуса документов ЦЦК

В этой главе мы приступаем к непосредственной разработке и тестированию информационной системы отслеживания статуса документов для ЦЦК. Эта часть работы представляет собой практическую реализацию теоретических знаний и анализа требований, описанных в предыдущих главах. В ней мы сосредоточимся на ключевых аспектах разработки, создание моделей данных, разработку интерфейса пользователя и тестирование всей системы.

3.1 Реализация информационной системы отслеживания документов

В данном разделе рассмотрим процесс реализации информационной системы отслеживания статуса документов для ЦЦК с использованием фреймворка Django. Модели данных являются центральным элементом любого Django-проекта, так как они определяют структуру данных, которую система будет хранить и обрабатывать. Модели в Django представляют собой подклассы класса django.db.models.Model, где каждый атрибут класса представляет собой поле модели, соответствующее колонке в таблице базы данных. Модели определяются в файле models.py каждого.

В нашем проекте были созданы следующие модели: Документ(Document), Акт (Act), Поставщик (Provider), Пользователь (User), Тип договора (ContractType), Тип акта (ActType), История договора (ContractHistory), История Акта (ActHistory). Все эти модели определены в файле models.py и используются для создания, чтения, обновления и удаления записей в базе данных. Рассмотрим реализацию нескольких из них.

Модель Договор (Contract) в Django представляет собой таблицу в базе данных, которая хранит информацию о договорах, заключенных с

поставщиками. Эта модель играет ключевую роль в информационной системе, позволяя отслеживать статус каждого договора на протяжении всего жизненного цикла. На рисунке 7 показано как выглядит модель договор.

```python
class Contract(models.Model):
    class Status(models.TextChoices):
        DRAFT = ("DRAFT", "Черновик")
        APPROVED_BY_MANAGER = (
            "APPROVED_BY_MANAGER",
            "Согласовал руководитель",
        )
        TRANSFERRED_TO_LEGAL_DEPARTMENT = (
            "TRANSFERRED_TO_LEGAL_DEPARTMENT",
            "Передан в юридический отдел",
        )
        SIGNED_TSU = ("SIGNED_TSU", "Подписан ТГУ")
        TRANSFERRED_TO_MATERIAL_DEPARTMENT = (
            "TRANSFERRED_TO_MATERIAL_DEPARTMENT",
            "Передан в материальный отдел",
        )
        REGISTERED = (
            "REGISTERED",
            "Зарегистрирован",
        )
        SENT_TO_PROVIDER = (
            "SENT_TO_PROVIDER",
            "Отправлен поставщику",
        )
        APPROVED_BY_PROVIDER = (
            "APPROVED_BY_PROVIDER",
            "Согласован поставщиком",
        )
    created_at = models.DateTimeField(verbose_name="Дата создания", auto_now_add=True)
    updated_at = models.DateTimeField(verbose_name="Дата обновления", auto_now=True)
    name = models.CharField(verbose_name="Наименование", max_length=250)
    contract_type = models.ForeignKey(
        to=ContractType,
        verbose_name="Тип",
        on_delete=models.SET_NULL,
        null=True,
    )
    status = models.CharField(
        verbose_name="Статус",
        max_length=250,
        choices=Status.choices,
        default=Status.DRAFT,
    )
    number = models.CharField(verbose_name="Номер", max_length=100, blank=True, null=True)
    registration_date = models.DateField(verbose_name="Дата регистрации", blank=True, null=True)
    provider = models.ForeignKey(
        to=Provider,
        verbose_name="Поставщик",
        on_delete=models.SET_NULL,
        null=True,
    )
    user = models.ForeignKey(
        to=User,
        on_delete=models.SET_NULL,
        verbose_name="Сотрудник ЦЦК создавший договор",
        blank=True,
        null=True,
    )

    class Meta:
        verbose_name = "Договор"
        verbose_name_plural = "Договоры"

    def __str__(self):
        return f"Договор: {self.name}"
```

Рисунок 7 – Модель Contract

Структура модели Contract:

- статусы договора (Status): Модель Contract использует внутренний класс Status, который определяет возможные статусы договора. Это позволяет легко управлять и отображать статус договора в пользовательском интерфейсе. Статусы включают "Черновик", "Согласовал руководитель", "Передан в юридический отдел", "Подписан ТГУ", "Передан в материальный отдел", "Зарегистрирован", "Отправлен поставщику" и "Согласован поставщиком";
- дата создания (created_at): поле содержит параметр auto_now_add, которая позволяет зафиксировать момент сохранения первой записи в таблицу. Поле автоматически заполняются системой;
- дата обновление (updated_at): поле заполняется системой автоматически при создании и последующим обновление записи о договоре, что позволяет отслеживать история изменений;
- наименование (name): поле для хранения наименования договора;
- тип договора (contract_type): связь с моделью ContractType, которая определяет тип договора. Это позволяет классифицировать договоры по различным категориям;
- статус (status): поле, которое выбирает статус из предопределенного списка статусов, определенных в классе Status;
- номер (number): поле для хранения номера договора;
- дата регистрации (registration_date): Дата, когда договор был зарегистрирован;
- поставщик (provider): связь с моделью Provider, которая представляет поставщика, с которым заключен договор;
- сотрудник ЦЦК (user): связь с моделью User, которая представляет сотрудника ЦЦК, создавшего договор. Это позволяет отслеживать, кто именно участвовал в процессе заключения договора;

Модель Истории договора (ContractHistory) предназначена для отслеживания истории изменений статуса договора. Благодаря этой модели можно отслеживания эволюции статусов договора во времени, позволяя детально аудиторить и анализировать события жизненного цикла договора. Она наследуется от абстрактной модели BaseHistory, что позволяет использовать общую структуру для различных типов историй изменений в приложении.

На рисунке 8 показана модель ContractHistory.

```python
class BaseHistory(models.Model):
    created_at = models.DateTimeField(verbose_name="Дата создания", auto_now_add=True)
    user = models.ForeignKey(
        to=User,
        on_delete=models.CASCADE,
        verbose_name="Пользователь",
        null=True,
    )

    class Meta:
        abstract = True


3 usages
class ContractHistory(BaseHistory):
    contract = models.ForeignKey(
        to=Contract,
        on_delete=models.CASCADE,
        verbose_name="Договор",
    )
    old_status = models.CharField(
        max_length=255,
        choices=Contract.Status.choices,
        verbose_name="Старый статус",
        null=True,
        blank=True,
    )
    new_status = models.CharField(
        max_length=255,
        choices=Contract.Status.choices,
        verbose_name="Новый статус",
        null=True,
        blank=True,
    )

    class Meta:
        verbose_name = "История договора"
        verbose_name_plural = "Истории договора"

    def __str__(self):
        return f"История договора: {self.contract}"
```

Рисунок 8 - Модель ContractHistory и BaseHistory

Структура модели ContractHistory:

- договор (Contract): устанавливает связь с моделью Contract, позволяя отслеживать, какие изменения происходили с конкретным контрактом. Аргумент on_delete=models.CASCADE обеспечивает, что при удалении контракта все связанные записи истории изменений также будут удалены, что помогает поддерживать целостность данных;
- старый статус (old_status) и новый статус (new_status): эти поля типа CharField используются для хранения информации о старом и новом статусе договора в момент изменения. Они используют параметр choices для выбора данных из предопределенного списка, что обеспечивает консистентность данных и уменьшает вероятность ошибок;
- дата создание (created_at): наследуется от базовой модели BaseHistory и автоматически заполняется текущей датой и временем при создании записи;
- пользователь (user): устанавливает связь с моделью User, указывающая, кто внес изменения в статус договора.

Класс ContractAdmin представляет собой специализированное управление административной панели для модели Contract. Он наследуется от admin.ModelAdmin, что позволяет настраивать представление и поведение этой модели в административном интерфейсе. Подробное описание данного класса:

- autocomplete_fields: автоматическое заполнение полей для contrcat_type и user, что упрощает выбор этих значений в интерфейсе администратора;
- list_display: определяет поля, которые будут отображаться в списке контрактов на странице списка в административной панели. Включает в себя название, тип договора, статус, номер и пользователя;

– list_filter: позволяет фильтровать список договоров по типу договора, статусу, дате регистрации и поставщику;
– readonly_fields: список полей, которые отображаются в форме редактирования, но не могут быть изменены пользователем. Включает в себя даты создания и обновления, а также пользователя создавший запись;
– search_fields: поля, по которым можно осуществлять поиск договора в административной панели;
– inlines: использует HistoryContractInline для отображения истории изменений статуса контракта непосредственно на странице договора в административной панели.

Техническая реализация административной панели приведена в Приложении Б.

Метод save_model в классе ContractAdmin является ключевым элементом для управления процессом сохранения объектов модели Contract. Этот метод переопределяет стандартное поведение сохранения, позволяя выполнять дополнительные операции перед или после сохранения объекта.

Аргументы метода, следующие:

– request: объект HttpRequest, который содержит информацию о текущем HTTP-запросе.
– obj: объект модели, который должен быть сохранен;
– form: форма, используемая для сохранения объекта;
– change: булева переменная, указывающая, является ли текущая операция изменением существующего объекта или созданием нового.

Класс ContractAdmin представлен на рисунке 9.

```python
@admin.register(Contract)
class ContractAdmin(admin.ModelAdmin):
    autocomplete_fields = ("contract_type", "user")
    list_display = (
        "name",
        "contract_type",
        "status",
        "number",
        "user",
    )
    list_filter = (
        ContractTypeFilter,
        "status",
        "registration_date",
        ProviderFilter,
    )
    readonly_fields = ("created_at", "updated_at", "user")
    search_fields = ("name", "number", "provider__name", "user__username")
    inlines = (HistoryContractInline,)

    def has_delete_permission(self, request, obj=None):
        return False

    def save_model(self, request, obj, form, change):
        try:
            instance_before_save = Contract.objects.get(id=obj.id)
        except Contract.DoesNotExist:
            instance_before_save = None
        old_status = instance_before_save.status if instance_before_save else None

        if not change:
            obj.user = request.user
        super().save_model(request, obj, form, change)
        ContractHistory.objects.create(
            user=request.user,
            contract=obj,
            old_status=old_status,
            new_status=obj.status,
        )
```

Рисунок 9 - Класс ContractAdmin

Логика метода:

– получение предыдущего состояния объекта: Метод пытается получить объект Contract с тем же id, что и текущий объект (obj). Если такой объект не найден (что может произойти при создании нового объекта), переменная instance_before_save устанавливается в None. В противном случае, извлекается статус этого объекта и сохраняется в переменной old_status;

- назначение автора: если текущая операция не является созданием нового объекта (то есть change равно False), то пользователь, совершивший запрос (request.user), назначается на объект obj как его автор;
- вызов родительского метода save_model: перед сохранением объекта вызывается метод super().save_model(request, obj, form, change), который обеспечивает сохранение объекта в базе данных с использованием стандартного механизма Django;
- создание записи истории: после успешного сохранения объекта создается новая запись в модели ContractHistory. Эта запись содержит информацию о пользователе, который совершил изменение (request.user), контракте, который был изменен (obj), старом статусе (old_status), если он был получен, и новом статусе контракта (obj.status).

В ходе разработки информационной системы отслеживания статуса документов ЦЦК были реализованы ключевые модели, обеспечивающие основу для эффективного управления документами.

3.2 Тестирование информационной системы отслеживания документов

Тестирование является неотъемлемой частью процесса разработки информационной системы отслеживания статуса документов ЦЦК. Оно помогает обеспечить, что система работает надежно и безопасно, что все функции выполняются в соответствии с требованиями, и что пользователи могут легко и эффективно управлять системой.

В рамках разработки информационной системы отслеживания статуса документов для ЦЦК была реализована административная панель, предназначенная для управления всеми аспектами работы системы. Эта панель предоставляет широкий спектр возможностей для управления данными

и процессами, связанными с договорами и актами. Она включает в себя функции, которые позволяют не только эффективно управлять документами, но и оптимизировать процессы, связанные с их жизненным циклом.

Административная панель предусматривает систему авторизации, которая позволяет управлять доступом к различным функциям и данным. Django предоставляет мощную систему аутентификации и авторизации, которая позволяет управлять учетными записями пользователей, группами, разрешениями и сеансами на основе файлов cookie. Сотрудники ЦЦК могут получить доступ к панели после успешной аутентификации, что обеспечивает безопасность и контроль над доступом к информации. После создание учетной записи сотрудником, необходимо присвоить им группу разрешений "Сотрудник ЦЦК". Группы используются для категоризации пользователей, а разрешения определяют, какие действия пользователи могут выполнять в системе. Данная группа содержит разрешений на чтение, добавление, изменение и удаление договоров, типы договоров, актов, поставщиков и т.д.

На рисунке 10 показан процесс авторизации нового пользователя как сотрудника ЦЦК.

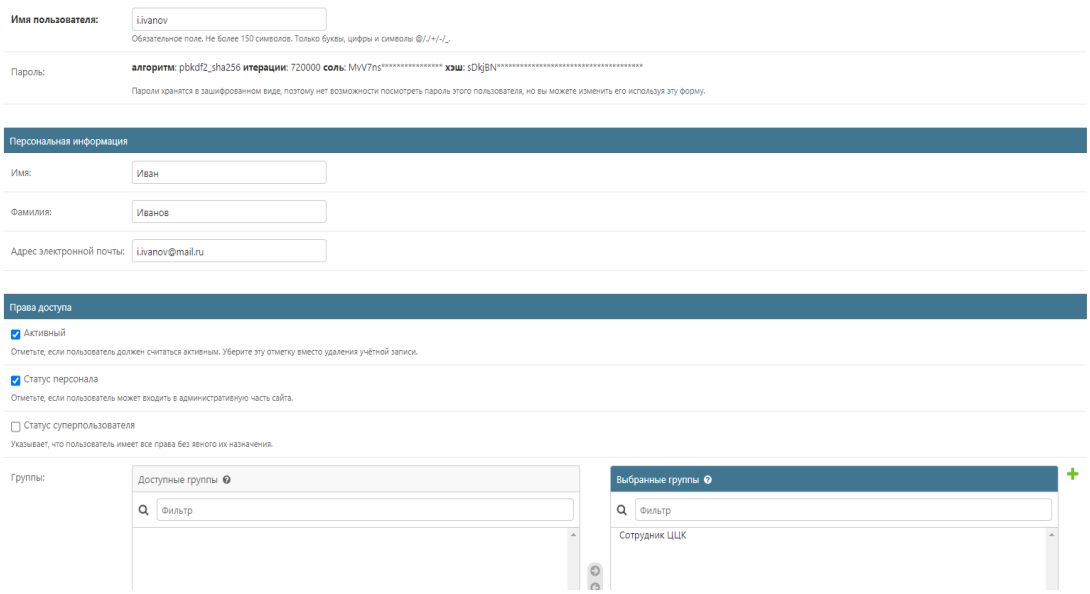


Рисунок 10 –Авторизации нового сотрудника ЦЦК

Сотрудник ЦЦК имеет возможность просматривать список всех договоров и актов, которые находятся в системе. Это включает в себя детальную информацию о каждом документе, такую как номер договора, дата регистрация, тип договора, сотрудник создавший договор и статус. Просмотр списка позволяет быстро ознакомиться с текущим состоянием документов и принимать решения о дальнейших действиях.

На рисунке 11 представлен список договор и список возможных фильтраций.

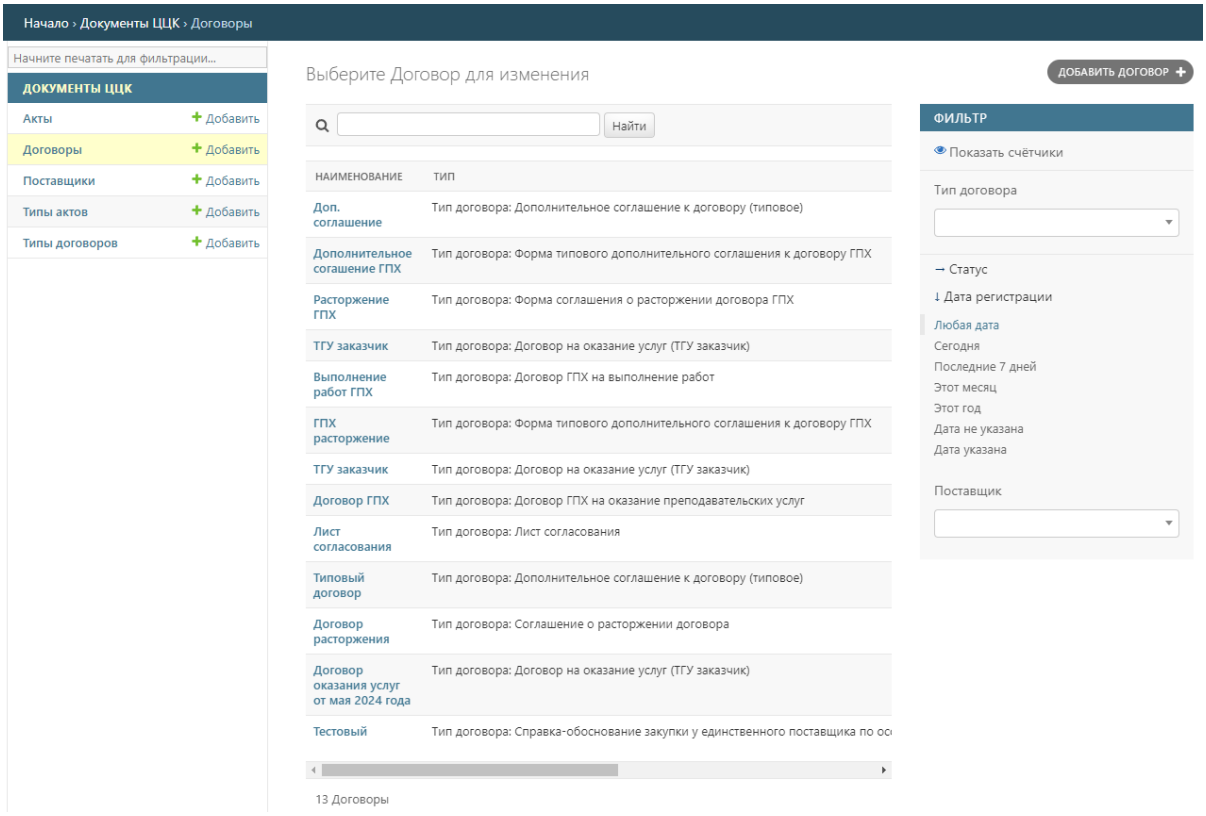


Рисунок 11 – Список договоров

Для удобства работы с большим количеством документов система предоставляет функционал фильтрации списка договоров и актов. Сотрудник ЦЦК может использовать различные параметры для фильтрации, такие как статус документа, дата регистрация, тип договора и поставщик. Фильтрация

позволяет сосредоточиться на конкретных документах, что значительно упрощает процесс поиска нужной информации.

Дополнительно к фильтрации, система предлагает функционал поиска по списку договоров и актов. Сотрудник ЦЦК может использовать название, номер, название поставщика и сотрудника, который создал договор, для поиска документов, что позволяет быстро находить необходимую информацию среди большого объема данных.

На рисунке 12 показана фильтрованный список актов по статусу и договору.

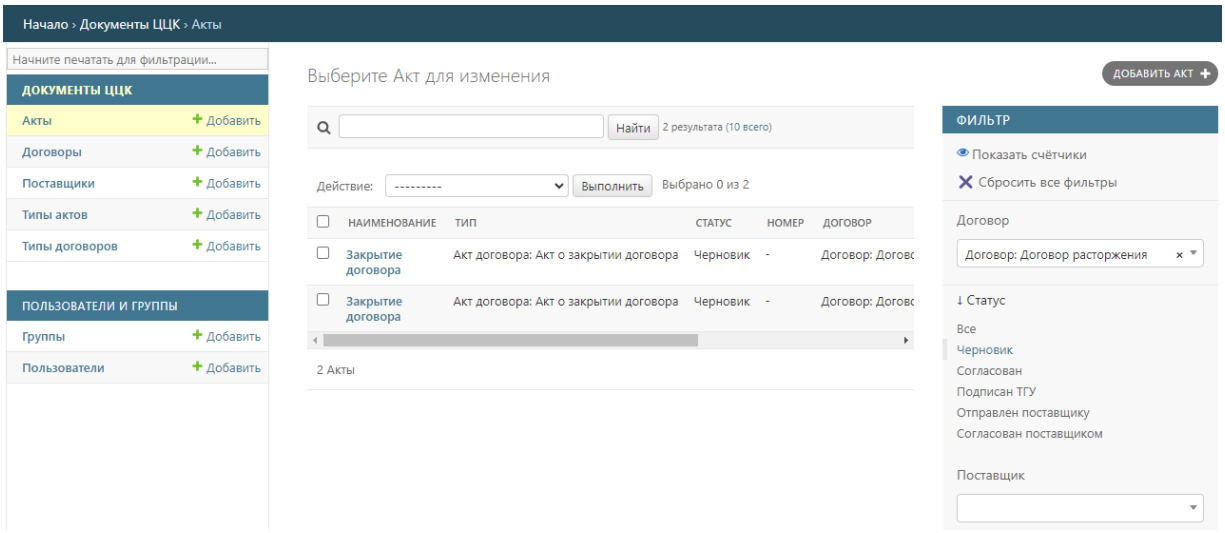


Рисунок 12– Фильтрованный список актов

Одной из ключевых функций системы является возможность добавления новых договоров. Сотрудник ЦЦК имеет возможность добавлять новые договоры в систему. Это включает в себя создание нового объекта договора, заполнение необходимых полей, таких как номер договора, дата регистрации, статус договора, тип договора и выбор поставщика. При добавлении нового договора система автоматически присваивает дату создания и устанавливает пользователя, который является создателем договора. После добавления договора информация о нем автоматически сохраняется в базе данных, обеспечивая актуальность и целостность данных.

На рисунке 13 представлена окно создания нового договора.

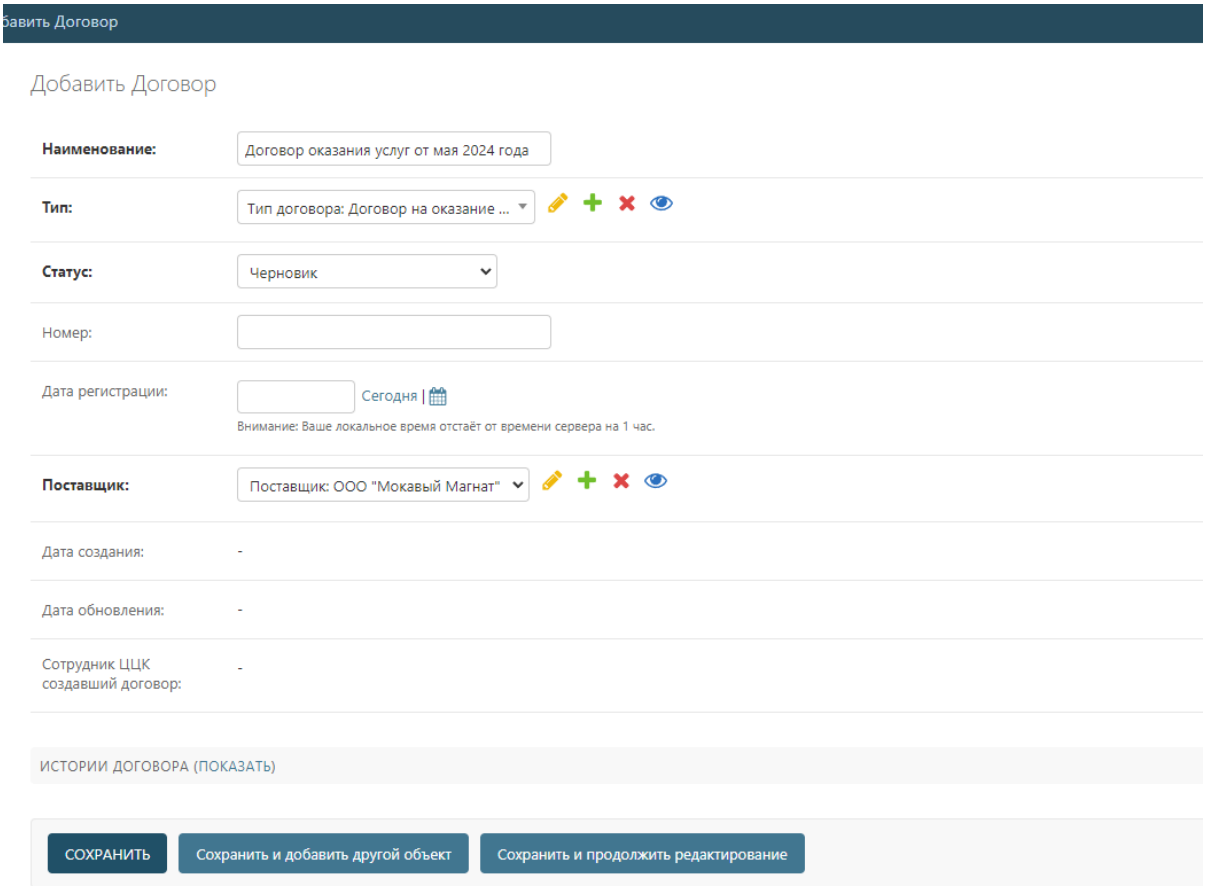


Рисунок 13 – Добавление нового договора

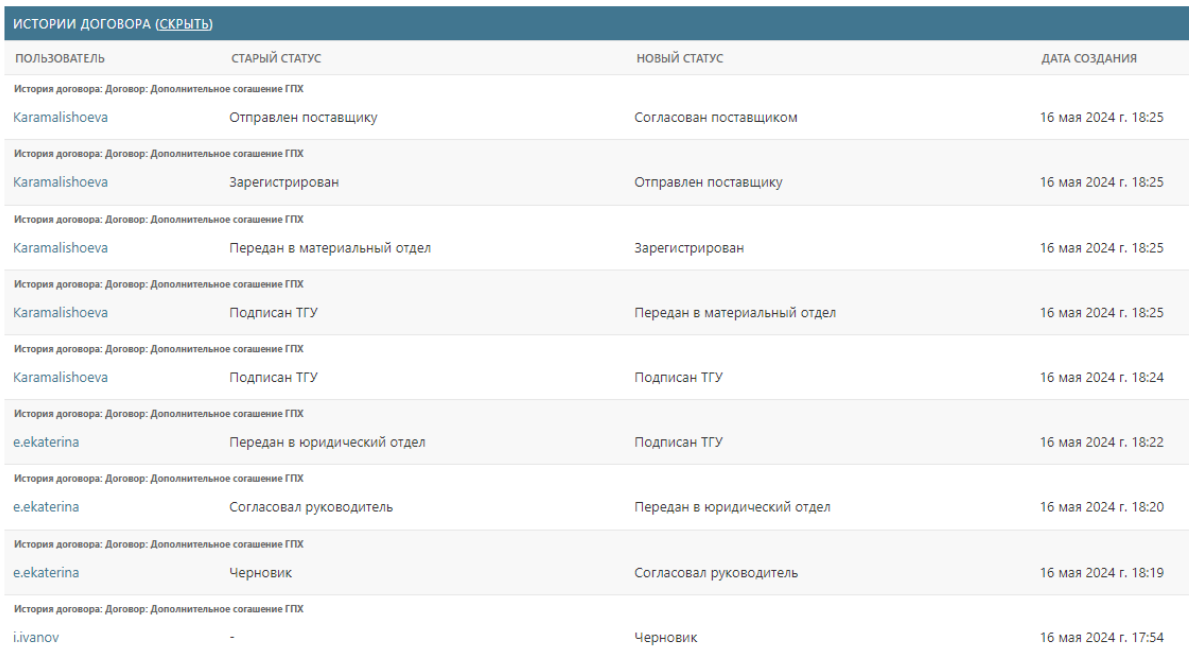
Один из основных требований к системе отслеживания статуса документов для ЦЦК, заключался в обеспечении возможности изменения статуса договора и отслеживания этого статуса в реальном времени. Это требование было направлено на обеспечение гибкости и динамичности управления договорами, а также на повышение прозрачности и контроля над процессом их обработки. Статусы отражают различные этапы жизненного цикла договора.

Сотрудник ЦЦК имеет возможность изменять статус договоров, что позволяет отражать текущее состояние каждого договора в системе. При изменении статуса договора система автоматически сохраняет историю изменений статуса договора, включая данные пользователя, который изменил

статус, старый статус, новый статус и время изменения. Это обеспечивает полную прозрачность и возможность отслеживания всех изменений, произошедших с договором. При изменении договора дата обновления автоматически присваивается. Это обеспечивает, что каждый договор имеет четкую временную метку последнего обновления, что критически важно для отслеживания истории изменений и управления документами.

Для удобства использования и быстрого доступа к информации, история договора отображается непосредственно на странице каждого договора (рисунок 14).

На рисунке 14 представлена история изменения статуса договора.



ИСТОРИИ ДОГОВОРА (СКРЫТЬ)

ПОЛЬЗОВАТЕЛЬ	СТАРЫЙ СТАТУС	НОВЫЙ СТАТУС	ДАТА СОЗДАНИЯ
История договора: Договор: Дополнительное соглашение ГПХ			
Karamalishoeva	Отправлен поставщику	Согласован поставщиком	16 мая 2024 г. 18:25
История договора: Договор: Дополнительное соглашение ГПХ			
Karamalishoeva	Зарегистрирован	Отправлен поставщику	16 мая 2024 г. 18:25
История договора: Договор: Дополнительное соглашение ГПХ			
Karamalishoeva	Передан в материальный отдел	Зарегистрирован	16 мая 2024 г. 18:25
История договора: Договор: Дополнительное соглашение ГПХ			
Karamalishoeva	Подписан ТГУ	Передан в материальный отдел	16 мая 2024 г. 18:25
История договора: Договор: Дополнительное соглашение ГПХ			
Karamalishoeva	Подписан ТГУ	Подписан ТГУ	16 мая 2024 г. 18:24
История договора: Договор: Дополнительное соглашение ГПХ			
e.ekaterina	Передан в юридический отдел	Подписан ТГУ	16 мая 2024 г. 18:22
История договора: Договор: Дополнительное соглашение ГПХ			
e.ekaterina	Согласовал руководитель	Передан в юридический отдел	16 мая 2024 г. 18:20
История договора: Договор: Дополнительное соглашение ГПХ			
e.ekaterina	Черновик	Согласовал руководитель	16 мая 2024 г. 18:19
История договора: Договор: Дополнительное соглашение ГПХ			
i.ivanov	-	Черновик	16 мая 2024 г. 17:54

Рисунок 14 – История изменения статуса договора

Это позволяет сотрудникам ЦЦК легко просматривать всю историю изменений договора, включая даты и пользователей, которые внесли изменения, а также старые и новые статусы. Название сотрудников, указанных в истории изменений договора, является кликабельными. При нажатии на них пользователь перенаправляется на страницу с подробной информацией о

данном сотруднике. Это обеспечивает не только удобство навигации по системе, но и позволяет получить дополнительную информацию о каждом участнике процесса изменения договора.

Такой подход к отображению истории изменений договора значительно упрощает процесс управления документами, позволяя быстро ориентироваться в истории изменений и принимать обоснованные решения.

Сотрудники ЦЦК имеют возможность добавлять данные о поставщиках в систему. Это включает в себя создание новых записей о поставщиках, заполнение необходимых полей, таких как название компании, контактная информация, тип поставщика и другая важная информация.

При создании нового договора или акта сотрудник ЦЦК может выбрать поставщика из списка зарегистрированных в системе. Это позволяет быстро и легко связывать документы с конкретными поставщиками, что упрощает процесс управления взаимоотношениями с партнерами и обеспечивает актуальность информации о поставщиках в контексте конкретных сделок.

На рисунке 15 показана процесс добавления нового поставщика в систему.

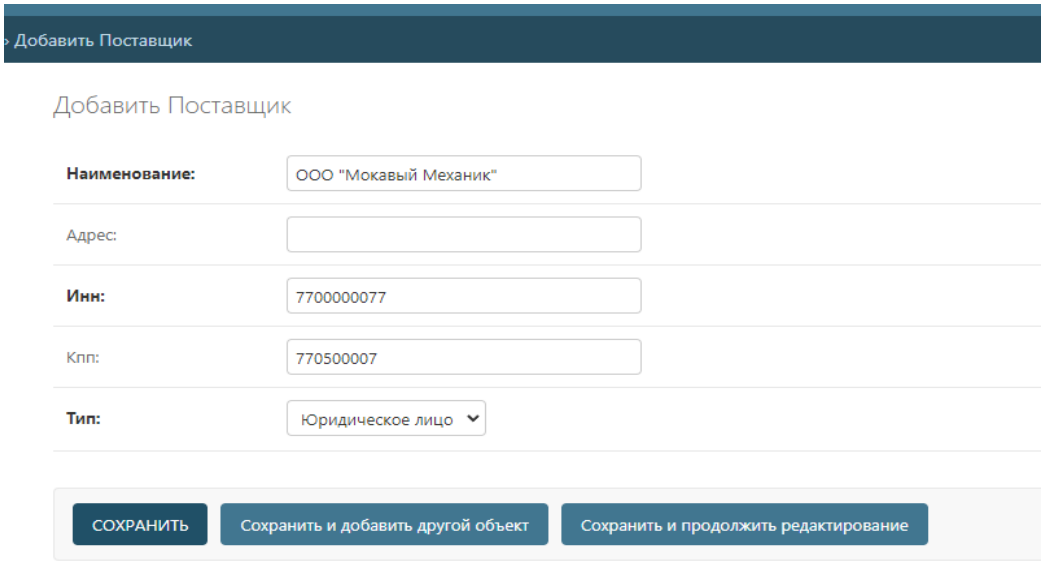


Рисунок 15 – Добавление нового поставщика

Другая ключевая функция системы является возможность добавления новых актов. Сотрудники ЦЦК имеют возможность добавлять новые акты в систему. При создании акта сотруднику необходимо выбирать договор, по которому будет создан акт, что позволяет установить прямую связь между актом и конкретным договором. Также в процессе создания акта необходима заполнить такие поля, как номер акта, тип акта, статус и выбрать поставщика. При создании акта автоматически устанавливается дата создания и пользователь, который создал акт. Это обеспечивает точность и целостность данных.

На рисунке 16 показана процесс создания нового акта.

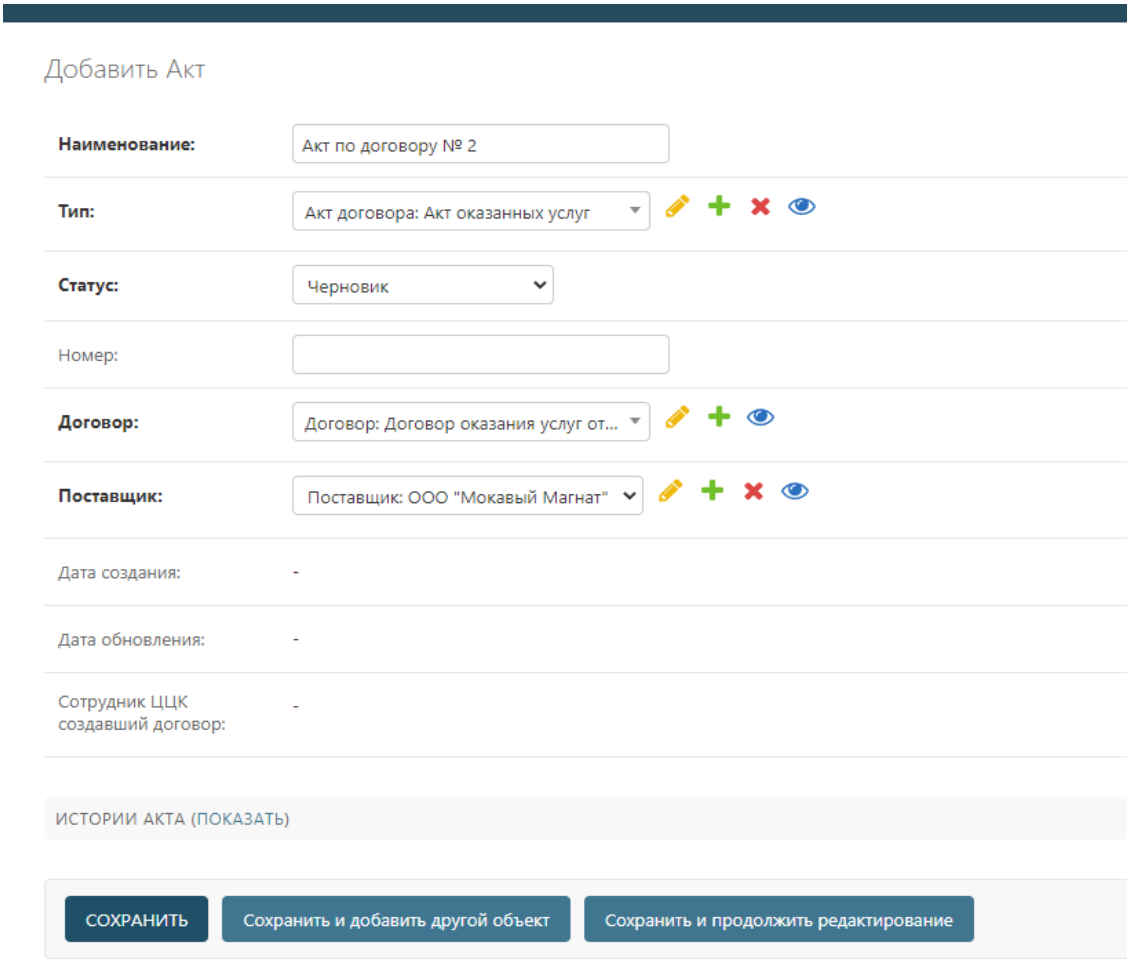


Рисунок 16 – Создание нового акта

После создания акта сотрудники ЦЦК могут вносить изменения в его содержание или статус. Это позволяет адаптировать акты под меняющиеся условия и требования, а также отражать текущее состояние каждого акта в системе. В случае обнаружения несоответствий или необходимости внесения дополнительных изменений, можно редактировать данные акта.

Также при изменении статуса акта, система автоматически добавляет запись в историю изменений. Эта запись включает в себя данные о пользователе, который внес изменения, старый статус акта, новый статус и время изменения. Хранение истории изменений позволяет отслеживать жизненный цикл акта.

Вывод по третьей главе

Тестирование разработанной ИС для отслеживания статуса документов ЦЦК, подтвердило высокую эффективность и полезность разработанной системы. Она обеспечивает удобство использования, улучшает управление документами, обеспечивает безопасность и конфиденциальность данных, а также расширяет аналитические возможности. Сотрудники, использующие эту систему, получают возможность более глубоко управлять документами, отслеживать их статусы и жизненный цикл, что в итоге приводит к улучшению работы всей организации.

Заключение

В данной бакалаврской работе была поставлена цель – разработать информационную систему для отслеживания статуса документов ЦЦК.

В ходе выполнения бакалаврской работы был проведен детальный анализ требований к системе в результате чего позволила выявить ключевые функциональные и технические требования к ИС, после чего был сделан выбор технологического стека для ее разработки. В качестве основной технологической платформы был выбран Django. В дополнение к Django, для обеспечения надежности и производительности данных, было принято решение использовать PostgreSQL в качестве базы данных. Эта комбинация технологий обеспечивает высокую степень масштабируемости и безопасности данных.

В результате была разработана ИС с учетом специфики работы с документами ЦЦК. В процессе разработки особое внимание уделялось удобству использования чтобы максимизировать его принятие сотрудниками. Были проведены тесты производительности и удобства использования, которые позволили сделать вывод о том, что система успешно выполняет свои функции, обеспечивая удобный интерфейс для создания, просмотра и управления документами.

Сотрудники ЦЦК теперь могут создавать договоры и акты, видеть список актов и договоров, фильтровать их по различным параметрам, изменять статус договора и акта, видеть историю изменения статуса договора и акта и многое другое.

В заключение, разработанная информационная система для отслеживания статуса документов ЦЦК успешно справляется с поставленной задачей, предоставляя удобный и эффективный инструмент для управления документами. Она не только упрощает процессы, связанные с созданием, хранением и обработкой документов, но и способствует повышению общей продуктивности сотрудников ЦЦК.

Список используемой литературы

1. Агальцов В.П. Базы данных. В 2-х т. Т. 2. Распределенные и удаленные базы данных: Учебник. М.: ИД-ФОРУМ, НИЦ ИНФРА-М, 2019. - 272 с.

2. Агальцов В.П. Базы данных. Распределенные и удаление базы данных. Книга 2. М.: ИНФРА-М, 2024. 271 с.

3. Агальцов В.П. Базы данных. Распределенные и удаленные базы данных: учебник. М.: ИНФРА-М, 2021. 271 с.

4. Анатолий П. Python, Django и PyCharm для начинающих. СПб.: БХВ-Петербург, 2021. 464 с.

5. Антонио М. Django 4 в примерах: Пер. с англ. М.: ДМК-Пресс, 2023, 800 с.

6. Васильев А.Н. Программирование на Python в примерах и задачах. М.: Эксмо, 2022. 616 с.

7. Глушаков С.В., Ломотько Д.В. Базы данных. М.: Харьков: Фолио, 2019. 504 с.

8. Девид Т., Эндрю Х. Программист-прагматик. Пер. с англ. СПб.: ООО «Диалектика», 2020. 368 с.

9. Добряк П.В. Python. Красивые задачи для начинающих. СПб.: БХВ-Петербург, 2024. 352 с.

10. Дронов В.А. Django 2.1. Практика создания веб-сайтов на Python. СПб.: БХВ-Петербург, 2019. 672 с.

11. Дронов В.А. Django 4. Практика создания веб-сайтов на Python. СПб.: БХВ-Петербург, 2023. 800 с.

12. Ирв К. Объектно-ориентированное программирование с помощью Python. Пер. с англ. СПб.: БХВ-Петербург, 2024. 512 с.

13. Карпова И.П. Базы данных: Учебное пособие. СПб.: Питер, 2021. – 240 с.

14. Лусиану Р. Python. К вершинам мастерства: Пер. с англ. М.: МК Пресс, 2022. 898 с.

15. Макконелл С. Совершенный код. Мастер-класс. Пер. с англ. СПб.: БХВ, 2021. 896с.

16. Михал Я. Тарек З. Python. Лучшие практики и инструменты. 4-е изд. СПб.: Питер, 2024. 592 с.

17. Назаров С.В. Архитектура и проектирование программных систем. М.:ИНФРА-М, 2023. 374с.

18. Прохоренко Н.А., Дронов В.А. Python 3. Самое необходимое. СПб.: БХВ-Петербург, 2019. 608 с.

19. Харрингтон Д. Проектирование объектно-ориентированных баз данных. Пер. с англ. – М.: ДМК Пресс, 2021. – 272 с.

20. Эрик Э. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем. Пер. с англ. М: ООО «Вильямс», 2020, 448 с.

21. Antonio M. Mastering Django. Packt Publishing Ltd, 2019. 499 c.

22. Lucid S. Learning PostgreSQL. Packt Publishing Ltd, 2019. 299 c.

23. Salahaldin J., A. Volkov. Learning PostgreSQL 10, Second Edition. Birmingham: Packt Publishing Ltd., 2019. 433 p.

24. Schonig H. Mastering PostgreSQL 11, Second Edition: Expert techniques to build scalable, reliable, and fault-tolerant database applications. Packt Publishing, 2018. 446 p.

25. William S. Django for Beginners: Build websites with Python and Django. Manning Publications, 2020. 320 p.

Приложение А

Фрагмент кода создание сущностей

```
from django.contrib.auth.models import User
from django.db import models

class Contract(models.Model):
    class Status(models.TextChoices):
        DRAFT = ("DRAFT", "Черновик")
        APPROVED_BY_MANAGER = (
            "APPROVED_BY_MANAGER",
            "Согласовал руководитель",
        )
        TRANSFERRED_TO_LEGAL_DEPARTMENT = (
            "TRANSFERRED_TO_LEGAL_DEPARTMENT",
            "Передан в юридический отдел",
        )
        SIGNED_TSU = ("SIGNED_TSU", "Подписан ТГУ")
        TRANSFERRED_TO_MATERIAL_DEPARTMENT = (
            "TRANSFERRED_TO_MATERIAL_DEPARTMENT",
            "Передан в материальный отдел",
        )
        REGISTERED = (
            "REGISTERED",
            "Зарегистрирован",
        )
        SENT_TO_PROVIDER = (
            "SENT_TO_PROVIDER",
            "Отправлен поставщику",
        )
        APPROVED_BY_PROVIDER = (
            "APPROVED_BY_PROVIDER",
            "Согласован поставщиком",
        )
    created_at = models.DateTimeField(verbose_name="Дата создания", auto_now_add=True)
    updated_at = models.DateTimeField(verbose_name="Дата обновления", auto_now=True)
    name = models.CharField(verbose_name="Наименование", max_length=250)
    contract_type = models.ForeignKey(
        to=ContractType,
        verbose_name="Тип",
        on_delete=models.SET_NULL,
        null=True,
    )
    status = models.CharField(
        verbose_name="Статус",
        max_length=250,
        choices=Status.choices,
        default=Status.DRAFT,
    )
    number = models.CharField(verbose_name="Номер", max_length=100, blank=True,
null=True)
```

```
    registration_date = models.DateField(verbose_name="Дата регистрации", blank=True,
null=True)
    provider = models.ForeignKey(
        to=Provider,
        verbose_name="Поставщик",
        on_delete=models.SET_NULL,
        null=True,
    )
    user = models.ForeignKey(
        to=User,
        on_delete=models.SET_NULL,
        verbose_name="Сотрудник ЦЦК создавший договор",
        blank=True,
        null=True,
    )

class Meta:
        verbose_name = "Договор"
        verbose_name_plural = "Договоры"

    def __str__(self):
        return f"Договор: {self.name}"

class ActType(models.Model):
    name = models.CharField(verbose_name="Наименование", max_length=250)

    class Meta:
        verbose_name = "Тип акта"
        verbose_name_plural = "Типы актов"

    def __str__(self):
        return f"Акт договора: {self.name}"

class ActType(models.Model):
    name = models.CharField(verbose_name="Наименование", max_length=250)

    class Meta:
        verbose_name = "Тип акта"
        verbose_name_plural = "Типы актов"

    def __str__(self):
        return f"Акт договора: {self.name}"

class Act(models.Model):
    class Status(models.TextChoices):
        DRAFT = ("DRAFT", "Черновик")
        AGREED = ("AGREED", "Согласован")
        SIGNED_TSU = ("SIGNED_TSU", "Подписан ТГУ")
        SENT_TO_PROVIDER = (
            "SENT_TO_PROVIDER",
```

```
  "Отправлен поставщику",
        )
        APPROVED_BY_PROVIDER = (
            "APPROVED_BY_PROVIDER",
            "Согласован поставщиком",
        )
    created_at = models.DateTimeField(verbose_name="Дата создания", auto_now_add=True)
    updated_at = models.DateTimeField(verbose_name="Дата обновления", auto_now=True)
    name = models.CharField(verbose_name="Наименование", max_length=250)
    act_type = models.ForeignKey(
        to=ActType,
        verbose_name="Тип",
        on_delete=models.SET_NULL,
        null=True,
    )
    status = models.CharField(
        verbose_name="Статус",
        max_length=250,
        choices=Status.choices,
        default=Status.DRAFT,
    )
    number = models.CharField(verbose_name="Номер", max_length=100, blank=True,
null=True)
    contract = models.ForeignKey(
        to=Contract,
        verbose_name="Договор",
        on_delete=models.CASCADE,
    )
    provider = models.ForeignKey(
        to=Provider,
        verbose_name="Поставщик",
        on_delete=models.SET_NULL,
        null=True,
    )
    user = models.ForeignKey(
        to=User,
        on_delete=models.SET_NULL,
        verbose_name="Сотрудник ЦЦК создавший договор",
        blank=True,
        null=True,
    )

    class Meta:
        verbose_name = "Акт"
        verbose_name_plural = "Акты"

    def __str__(self):
        return f"Акт: {self.name}"
```

Приложение Б

Фрагмент кода конфигурации административной панели

```
from django.contrib import admin
from admin_auto_filters.filters import AutocompleteFilter

from .models import (
    Contract,
    ContractType,
    Act,
    ActType,
    ContractHistory,
    ActHistory,
    Provider,
)

class ContractTypeFilter(AutocompleteFilter):
    title = "Тип договора"
    field_name = "contract_type"

class ProviderFilter(AutocompleteFilter):
    title = "Поставщик"
    field_name = "provider"

class ContractFilter(AutocompleteFilter):
    title = "Договор"
    field_name = "contract"

class HistoryInlineBase(admin.TabularInline):
    extra = 0
    show_change_link = True
    classes = ("collapse",)
    can_delete = False
    readonly_fields = ("created_at",)
    ordering = ("-created_at",)

    def has_change_permission(self, *args, **kwargs):
        return False

    def has_add_permission(self, *args, **kwargs):
        return False

class HistoryContractInline(HistoryInlineBase):
    autocomplete_fields = (
        "user",
        "contract",
    )
    model = ContractHistory

@admin.register(Contract)
class ContractAdmin(admin.ModelAdmin):
    autocomplete_fields = ("contract_type", "user")
```

```
    list_display = (
        "name",
        "contract_type",
        "status",
        "number",
        "user",
    )
    list_filter = (
        ContractTypeFilter,
        "status",
        "registration_date",
        ProviderFilter,
    )
    readonly_fields = ("created_at", "updated_at", "user")
    search_fields = ("name", "number", "provider__name", "user__username")
    inlines = (HistoryContractInline,)

    def has_delete_permission(self, request, obj=None):
        return False

    def save_model(self, request, obj, form, change):
        try:
            instance_before_save = Contract.objects.get(id=obj.id)
        except Contract.DoesNotExist:
            instance_before_save = None
        old_status = instance_before_save.status if instance_before_save else None

        if not change:
            obj.user = request.user
        super().save_model(request, obj, form, change)
        ContractHistory.objects.create(
            user=request.user,
            contract=obj,
            old_status=old_status,
            new_status=obj.status,
        )


@admin.register(Act)
class ActAdmin(admin.ModelAdmin):
    autocomplete_fields = ("act_type", "user", "contract")
    list_display = (
        "name",
        "act_type",
        "status",
        "number",
        "contract",
        "user",
    )
    list_filter = (
```

```
    ContractFilter,
    "status",
    ProviderFilter,
)
readonly_fields = ("created_at", "updated_at", "user")
search_fields = ("name", "number", "provider", "user")
inlines = (HistoryActInline,)

def save_model(self, request, obj, form, change):
    try:
        instance_before_save = Act.objects.get(id=obj.id)
    except Act.DoesNotExist:
        instance_before_save = None
    old_status = instance_before_save.status if instance_before_save else None

    if not change:
        obj.user = request.user
    super().save_model(request, obj, form, change)
    ActHistory.objects.create(
        user=request.user,
        act=obj,
        old_status=old_status,
        new_status=obj.status,
    )
```