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

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

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

«Бизнес-информатика»
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка интеллектуальной системы управления платежами

Обучающийся А.Ю. Варнухов
(Инициалы Фамилия) (личная подпись)

Руководитель Н.Н. Казаченок
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант (ы) к.ф.н., доцент Н.В. Андрюхина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

## Аннотация

Бакалаврская работа выполнена на тему «Разработка интеллектуальной системы управления платежами».

Цель работы заключается в разработке интеллектуальной системы управления платежами для ООО «Проектное бюро Р1».

Во введении представлены: актуальность темы, объект и предмет исследования, методы исследования, цель работы, задачи для достижения поставленной цели, а также краткое содержание глав.

В первой главе рассмотрена характеристика предприятия, проведено моделирование бизнес-процесса управления платежами, определены требования к ИС, оценены существующие на рынке программные решения, выполнена постановка задачи на разработку ИС и построена модель бизнес-процесса TO-BE.

Во второй главе представлены варианты использования, разработаны диаграммы классов, деятельности, состояний, компонентов и развертывания, построены концептуальная и логическая модели данных, определены требования к программно-аппаратному обеспечению и представлены прототипы пользовательского интерфейса.

В третьей главе проанализированы технологии, платформы и средства разработки, выбрана СУБД, построена физическая модель данных, описано хранение данных и взаимодействие с ORM, представлены шаблоны для оформления, показана реализация контроля доступа, приведена информация по разработанным модулям системы, даны результаты экспериментов по поиску оптимального алгоритма машинного обучения, проведена оценка экономической эффективности.

Бакалаврская работа состоит из 76 страниц и включает 35 рисунков, 20 таблиц, 23 источника, 3 приложения.

## Abstract

The topic of the bachelor's work is The development of an intelligent payment management system.

The purpose of the work is to develop an intelligent payment management system for Project Bureau R1.

The introduction touches upon the relevance of the topic, the object and the subject of the research, the research methods, the purpose of the work, tasks to achieve the goal, as well as a summary of the chapters.

In the first chapter, the characteristics of the enterprise are considered, the modeling of the business process of managing payments is carried out, the requirements for the system are determined, the software solutions existing on the market are evaluated, the task for the development is set, and the TO-BE business process model is built.

The second chapter introduces use cases, class, activity, state, component and deployment diagrams, builds conceptual and logical data models, defines firmware requirements, and presents user interface prototypes.

The third chapter analyzes technologies, platforms and development tools, selects a DBMS, builds a physical data model, describes data storage and interaction with ORM, presents design templates, shows the implementation of access control, provides information on the developed system modules, gives the results of experiments to find the optimal algorithm of machine learning. The assessment of economic efficiency is also carried out.

The bachelor's work consists of 76 pages and includes 35 figures, 20 tables, 23 sources, 3 appendices.

## Оглавление

# Введение

Стабильность и развитие любого предприятия зависят от организации эффективного управленческого и финансового учета. Происходящие структурные изменения в экономике стимулируют многие предприятия заниматься сокращением издержек и повышением качества управления. Эффективное управление включает в себя задачу оптимизации затрат в рамках внутренних и внешних бизнес-процессов. Менеджмент предприятия должен иметь возможность оперативного контроля и управления финансовой ситуацией. Одним из способов решения этой задачи выступает разработка и внедрение автоматизированных информационных систем.

Целью настоящей работы является разработка интеллектуальной системы управления платежами для ООО «Проектное бюро Р1». Внедрение автоматизированной информационной системы позволит предприятию сократить трудовые и временные затраты, повысить скорость работы, а также улучшить контроль и оперативное управление.

Актуальность разработки обуславливается, во-первых, стратегией развития и планами по информатизации деятельности предприятия, а во-вторых, применением адаптивных алгоритмов машинного обучения, которые созданы и обучены с учетом уникальных особенностей предприятия.

Объектом исследования является деятельность финансовой службы ООО «Проектное бюро Р1». Предметом исследования является автоматизация бизнес-процесса управления платежами.

Выполнение поставленной цели включает следующие задачи:

- изучить организационную структуру предприятия и направления деятельности его подразделений;
- провести анализ бизнес-процесса управления платежами;
- выявить существенные недостатки, определить бизнес-требования и выполнить постановку задачи на разработку;
- выбрать и обосновать технологию проектирования;

– спроектировать функционал системы, сценарии взаимодействия, ключевые сущности, состояния, компоненты и модели данных;
– выбрать СУБД и технологическую платформу разработки;
– выполнить разработку информационной системы;
– оценить экономическую эффективность проекта информатизации.

В первой главе рассмотрена характеристика предприятия, обоснован выбор методологии моделирования, проведено моделирование бизнес-процесса управления платежами, определены требования к ИС, оценены существующие на рынке программные решения, выполнена постановка задачи на разработку ИС и построена модель бизнес-процесса TO-BE.

Во второй главе рассмотрены технологии проектирования, представлены варианты использования, описаны сценарии взаимодействия, приведены разработанные диаграммы классов, деятельности, состояний, компонентов, развертывания, а также построенные концептуальная и логическая модели данных, определены требования к программно-аппаратному обеспечению и продемонстрированы прототипы UI/UX.

В третьей главе проанализированы технологии, платформы и средства разработки, обоснован выбор СУБД, построена физическая модель данных, описано хранение данных в системе и взаимодействие с ORM, представлены используемые шаблоны для оформления, показана реализация контроля доступа, приведена информация по разработанным модулям системы, приведены результаты экспериментов по поиску оптимального алгоритма машинного обучения, проведена оценка экономической эффективности.

Результатом проделанной работы является разработанная интеллектуальная система управления платежами.

# Глава 1 Анализ бизнес-процесса управления платежами

## 1.1 Общая характеристика организации

Компания «Проектное бюро R1» учреждена в 2013 году в городе Екатеринбурге. Одним из ключевых стремлений основателей стало желание собрать коллектив увлеченных инновационными идеями единомышленников. На протяжении следующих лет компания продемонстрировала быстрый рост и заняла устойчивые позиции в своей сфере деятельности. Среди первых проектов компании можно отметить работы по проектированию ЖК Клевер-Парк и ЖК Малевич, а среди знаковых аэропорт Платов города Ростов-на-Дону и университет Шеффилда в Великобритании. На сегодняшний день компания выполняет широкий спектр работ связанных с проектированием зданий и сооружений. Профессиональная команда и наработанный опыт инженерных решений позволяет выполнять проекты любой степени сложности и масштабов. Центральный офис компании и большинство сотрудников располагаются в городе Екатеринбурге, также открыты офисы в Москве, Санкт-Петербурге, Новосибирске и Владивостоке. География ведения деятельности охватывает всю территорию Российской Федерации, а также некоторые зарубежные страны, в которых Проектное бюро R1 работает под брендом Twelve Architects & Masterplanners. Согласно данным из открытых источников по состоянию на 01.01.2021 год компания обладает собственным штатом в 110 человек и получила доход порядка 400 млн. рублей в год [14].

## 1.2 Направления деятельности и структурные подразделения

Компания «Проектное бюро R1» выполняет широкий перечень видов работ, включающий следующие основные направления:

- создание BIM моделей, концептуальное и мастер планирование;
- разработка проектной документации стадий ЭП, ПД, РД, РП;

– проектирование внутренних и наружных инженерных сетей.

Компания применяет линейно-функциональную модель организационной структуры. Организационная структура компании представлена на рисунке 1.

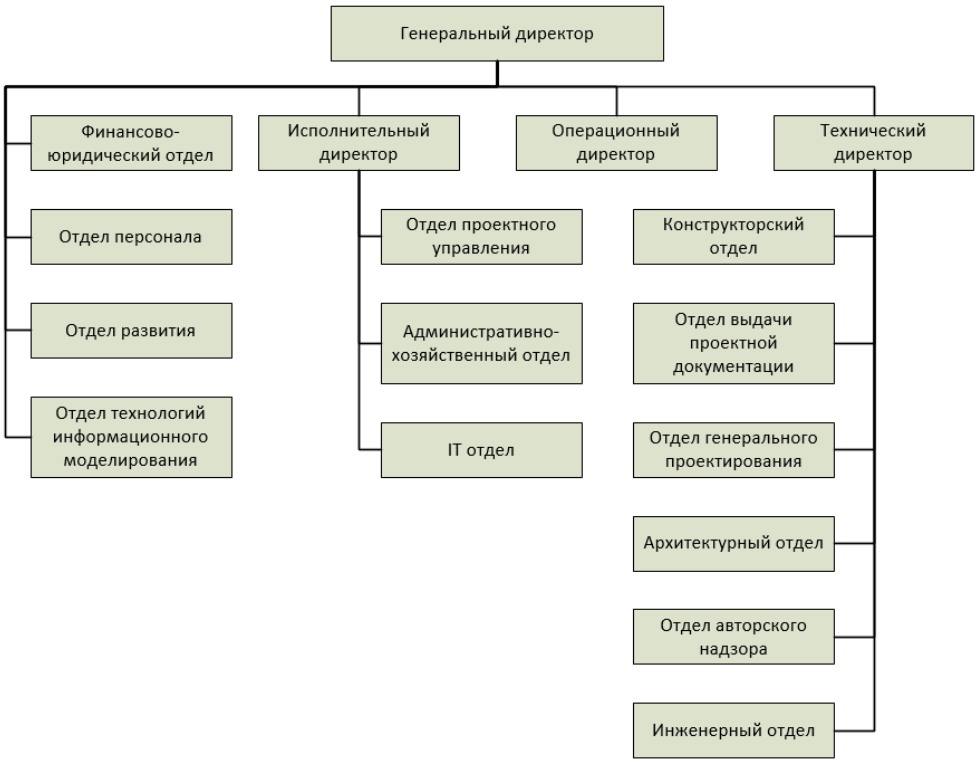


Рисунок 1 – Организационная структура предприятия

Такой принцип структурной организации позволяет обеспечить высокую эффективность управленческого потенциала сохраняя преимущества и устойчивость присущие линейным моделям [1].

### 1.3 Бизнес-процессы верхнего уровня

Согласно [3] бизнес-процессы компании для структурирования и анализа можно разделить на группы: основные, обеспечивающие, управления и развития. В таблице 1 приведены бизнес-процессы верхнего уровня предприятия.

Таблица 1 – Бизнес-процессы верхнего уровня

| Тип | Название |
|---|---|
| Основные бизнес-процессы | Обеспечение продаж и работа с клиентами<br>Технологическое проектирование<br>Архитектурное проектирование<br>Разработка проектной документации<br>Маркетинг<br>Проектная деятельность<br>Ведение авторского надзора |
| Обеспечивающие бизнес-процессы | Организация и хранение документации<br>Документооборот и договорная работа<br>Бухгалтерский и налоговый учет<br>Взаимодействие с субподрядчиками<br>Исполнение гарантийных обязательств<br>АХО |
| Бизнес-процессы управления | Управление коммуникациями<br>Управление персоналом<br>Управление закупками<br>Управление финансами<br>Корпоративное управление<br>Управление СМК |
| Бизнес-процессы развития | Обучение и развитие персонала<br>Разработка инноваций<br>Анализ и прогнозирования рыночной среды<br>Развитие и внедрение ИТ технологий |

К основным относятся процессы, которые обеспечивают создание ценности для заказчиков и дают возможность компании получать доход. Обеспечивающие процессы помогают предприятию вести деятельность. Бизнес-процессы управления позволяют вести эффективное управление компанией. К бизнес-процессам развития отнесена такая деятельность компании, которая направлена на получение запланированных результатов в некотором будущем.

### 1.4 Характеристика финансово-юридического отдела

Финансово-юридическим отделом руководит финансовый директор, который подчиняется напрямую генеральному директору компании. Основные виды работ, выполняемые подразделением:

– финансово-экономический анализ;
– разработка и контроль финансовой дисциплины;
– управление финансовыми потоками и ресурсами;
– ведение бухгалтерского и управленческого учета;
– формирование управленческой отчетности;
– разработка нормативной и правовой документации;
– договорная и претензионная работа;
– калькуляции и расчет ТЭО по проектам;
– экономическая оптимизация деятельности компании;
– кадровое делопроизводство и юридическая поддержка.

Организационная структура финансового-юридического отдела компании представлена на рисунке 2.

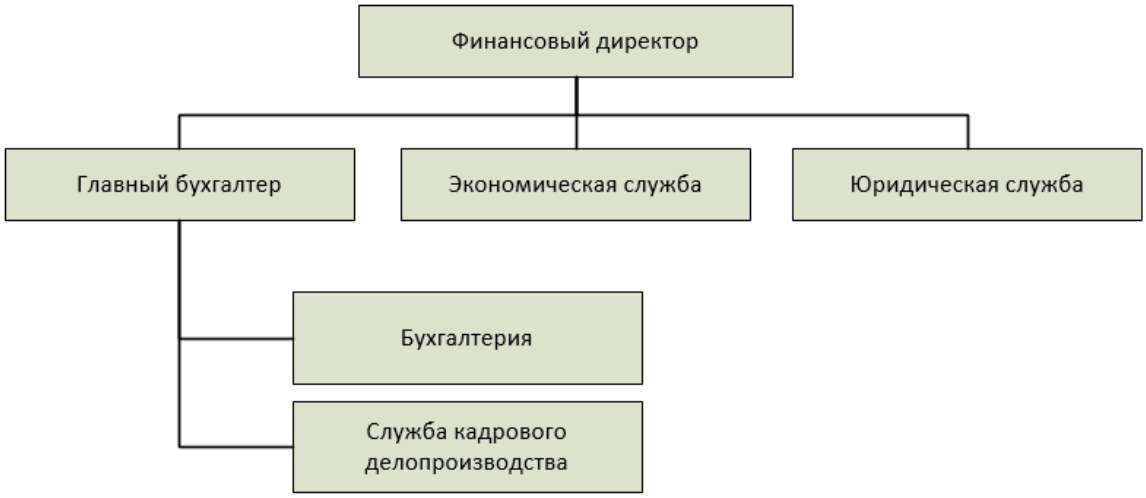


Рисунок 2 – Организационная структура финансового-юридического отдела

Для решения рабочих задач на рабочих станциях сотрудников установлены операционные системы семейства Microsoft Windows. Используемое аппаратное обеспечение соответствуют требованиям, которые предъявляются к офисному компьютеру, а именно: процессор не ниже Intel Core i3, 4 и более Гб оперативной памяти, жёсткий диск емкостью не менее

500 Гб. В связи с характером работы, связанным с просмотром и составлением сводных таблиц и графиков, в подразделении применяются широкоформатные мониторы с диагональю не менее 24 дюймов. Имеется доступ к сети Интернет. Перечень используемого программного обеспечения включает:

- Microsoft Office Excel и Microsoft Office Word;
- Google Sheets и Google Docs;
- Интернет браузер (Google Chrome, Mozilla Firefox, Microsoft Edge);
- Adobe Acrobat;
- 1С Предприятие и 1С Бухгалтерия;
- Контур-Экстерн и Контур-Фокус;
- Консультант Плюс и Гарант.

Проведенное обследование показало, что в подразделении не применяются специализированные информационные системы в части автоматизации рассматриваемой задачи.

### 1.5 Методология анализа и моделирования

Моделирование можно описать как процесс создания некоторой приближенной модели или образа, который удобно использовать при дальнейшем анализе и проектировании бизнес-процессов и информационных систем. При помощи методологий моделирования можно формализовать протекающие в настоящий момент бизнес-процессы, выполнить анализ и последующий реинжиниринг для улучшения ключевых бизнес-показателей компании, а также провести автоматизацию. На текущий момент существуют множество [15] методологий и нотаций, которые применяются для целей моделирования, проектирования и анализа: ARIS, BPMN, IDEF, UML, DFD и другие. Как правило, модели позволяют формализовать следующие аспекты анализируемых бизнес-процессов:

- перечень работ (функций), которые выполняются в рамках анализируемого бизнес-процесса;

– цепочку действий (шагов) внутри процесса и их взаимосвязи;
– владельца и ключевых участников бизнес-процесса;
– возникающие материальные и не материальные потоки данных;
– необходимые и используемые ресурсы и регламенты управления.

В процессе моделирования обычно разрабатываются функциональные, информационные и поведенческие модели. Функциональная модель позволяет отразить особенности работы бизнес-процесса, его взаимосвязи с другими процессами и в конечном итоге определяет, что и как необходимо выполнить для достижения конечного результата. Согласно [7] наиболее широкую распространённость для построения функциональных моделей получили методологии структурного анализа и проектирования IDEF0 и DFD.

Информационная модель, в отличие от функциональной, позволяет определить структуру и организацию информации в рамках анализируемых бизнес-процессов. Как правило, при построении информационных моделей проектируется будущее хранилище данных, а сам процесс проектирования может включать разработку концептуальной, логической и физической информационных моделей. Для моделирования информационных моделей успешно применятся метод моделирования IDEF1X [4].

Поведенческая модель отражает динамический аспект работы бизнес-процесса включая набор состояний, переходов и условий переходов между этими состояниями. Таким образом, поведенческая модель основывается и детализирует функциональную и информационную модель. Для построения поведенческих моделей могут применяться [13] теория массового обслуживания, сети Петри, модель конечного автомата, блок-схемы, методологии и нотации IDEF3 и BPMN.

Для обеспечения и поддержки процесса моделирования существуют такие CASE-средства, как: BPWin, ERWin, Rational Rose, Vantage Team Builder и другие. В таблице 2 представлены итоги проведенного опроса по выявлению предпочтений применения методологий анализа и проектирования согласно данным приведенным в [2].

Таблица 2 – Предпочтения по методологиям анализа и проектирования

| Методология | Результаты | |
|---|---|---|
| | Количество | Доля в % |
| Семейство IDEF и DFD | 594 | 50% |
| UML | 211 | 17% |
| ARIS | 160 | 13% |
| Безразлично | 150 | 12% |
| Другие | 95 | 8% |

Для выбора методологии необходимо учитывать различные факторы, которые относятся как к особенностям самой методологии, так и к специфике решаемой задачи в рамках реализуемого проекта информатизации. С точки зрения пользователя приведенных методологий важны такие качества, как: легкость применения, понятность, оригинальность и детерминированность.

Опираясь на [16] можно предложить следующие условия для выбора используемой методологии и нотации:

- содержат минимально необходимое количество репрезентативных элементов согласно предъявляемых требований;
- приемлемая скорость проектирования и низкие трудозатраты;
- доступность и легкость понимания полученных графических моделей всеми ключевыми участниками;
- наличие возможности описания сквозных бизнес-процессов;
- обеспеченность CASE-средствами для проектирования.

Учитывая масштаб, особенности и длительность проекта информатизации, а также вышеизложенное и руководствуясь [9] в настоящей работе будут применены семейство методологий IDEF, DFD и UML.

### 1.6 Модель бизнес-процесса AS-IS

Бизнес-процесс управления платежами это внутренний бизнес-процесс предприятия, который можно отнести к одному из важнейших в деятельности финансово-юридического отдела. Основные цели бизнес-процесса включают

контроль за расходованием денежных средства, планирование поступлений от контрагентов, а также анализ и регулирование платежной дисциплины.

Для проведения анализа и дальнейшей работы над задачей информатизации деятельности подразделения предприятия необходимо построить модель существующего бизнес-процесса «Как есть» (AS-IS). Такая модель позволяет определить участников процесса, формализовать порядок и особенности работы, потоки, ресурсы, документы и другую важную информацию [6].

Контекстная модель, построенная на основании анализа внутренних документов и опроса сотрудников предприятия, показана на рисунке 3.

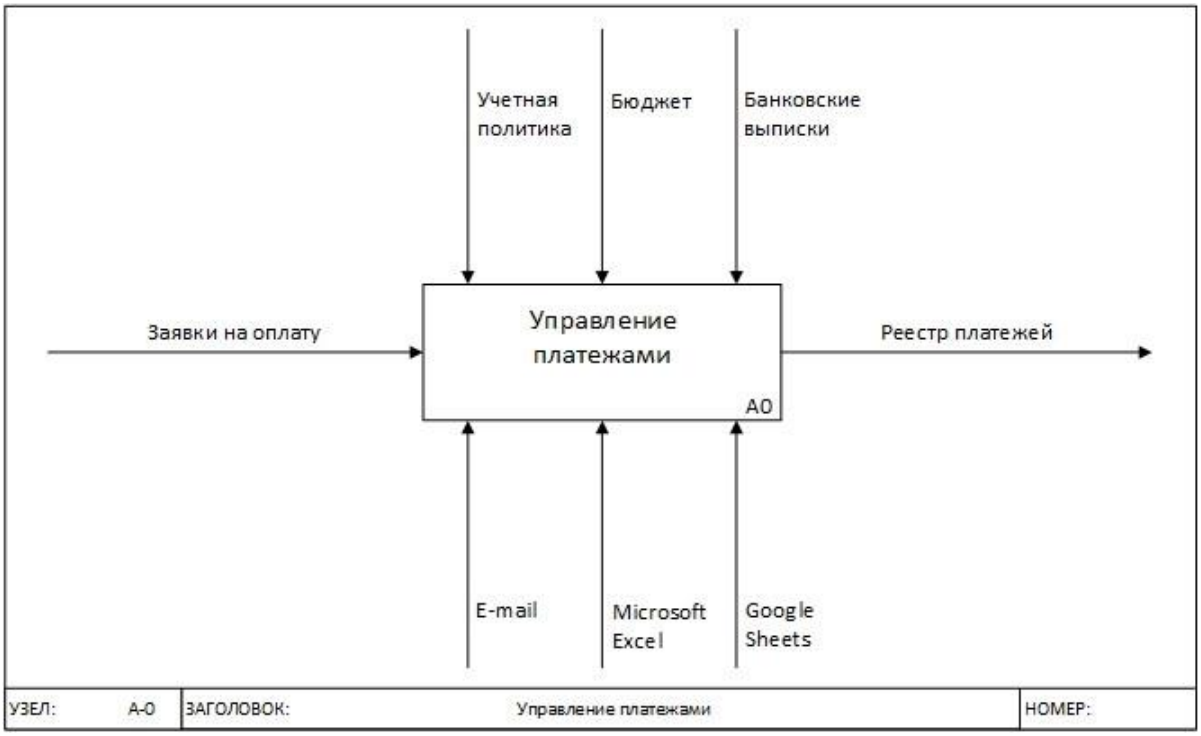


Рисунок 3 – Контекстная диаграмма модели процесса AS-IS

Моделирование выполнялось с точки зрения начальника экономического отдела с целью анализа существующего бизнес-процесса для поиска возможностей его усовершенствования посредством автоматизации. Диаграмма декомпозиции представлена на рисунке 4.

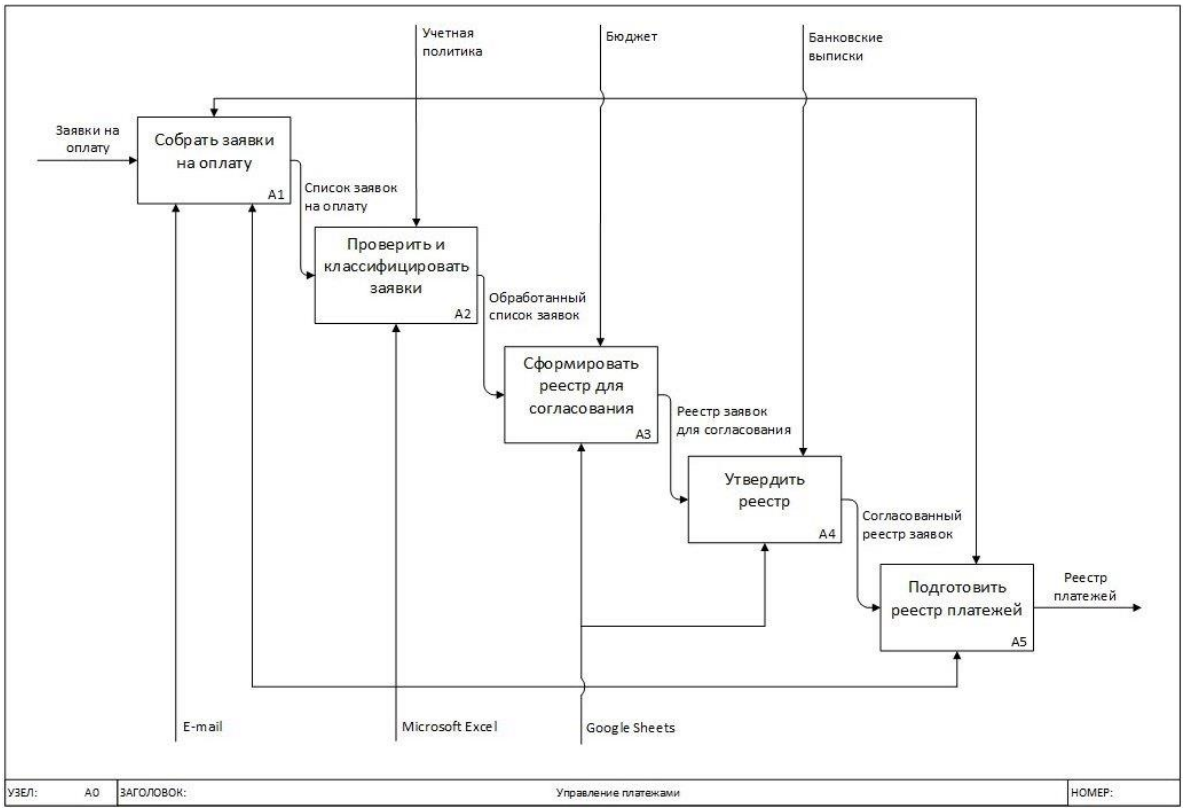


Рисунок 4 – Диаграмма декомпозиции модели процесса AS-IS

Бизнес-процесс начинается с подачи заявки сотрудником компании. Каждый сотрудник компании, в рамках своих должностных обязанностей, может подавать в финансово-юридический отдел заявки на оплату.

Для подачи заявок на оплату сотрудник должен отправить, заполненное в соответствие с учетной политикой, письмо на e-mail ответственного сотрудника экономического отдела. Сотрудник экономической службы выполняет сбор и проверку поступивших заявок. Все заявки проверяются на наличие необходимых документов, а контрагент на благонадежность. Каждой поступившей заявке на оплату должна быть присвоена статья затрат согласно утвержденному классификатору учёта.

Согласно платежной дисциплине, оплата поданных заявок и формирование реестра платежей происходит несколько раз в неделю. Собранный и обработанный список заявок передается начальнику экономической службы перед наступлением очередной даты формирования

реестра. Начальник экономической службы руководствуясь БДДС и БДР формирует реестр заявок для согласования. Поскольку всем участникам процесса утверждения требуется онлайн доступ и возможность совместного редактирования, то формирование реестра заявок осуществляется в облачной платформе Google Sheets.

Сформированный реестр заявок передается для проверки финансовому директору и после этого направляется на утверждение генеральному директору. Генеральный директор может одобрить, отклонить или отложить платеж по поданной заявке на оплату исходя из финансовой и производственной обстановки.

После согласования реестра заявок на оплату сотрудник экономической службы выполняет подготовку реестра платежей согласно действующей учетной политики в специальном формате при помощи Microsoft Excel. После этого реестр платежей распечатывается и передается в бухгалтерию для последующей оплаты.

### 1.7 Выявление существенных недостатков

На основании проведенного обследования и моделирования бизнес-процесса управления платежами среди прочих можно выделить следующие недостатки:

- подача заявок на оплату происходит с использованием e-mail, что вызывает появление ошибок, неточностей и несоответствий с учетной политикой;
- сотрудники экономической службы вынуждены тратить рабочее время на сбор, проверку и систематизацию заявок, которые были отправлены по e-mail;
- классификация заявок по статьям затрат выполняется вручную, что отнимает рабочее время и повышает риски ошибок;

– поскольку бизнес-процесс выполняется разными сотрудниками на разных рабочих станциях, то отсутствие на работе любого сотрудника увеличивает общие трудозатраты и потраченное время из-за необходимости получения доступа к данным сотрудника;
– из-за формирования реестра заявок для согласования в Microsoft Excel, а дальнейшей работы по утверждению реестра в Google Sheets, сотрудникам экономической службы приходится дублировать одинаковые данных в разных системах;
– требуются дополнительное время на обработку данных для преобразования реестра платежей в специальный формат для бухгалтерии, согласно учетной политики;
– использование нескольких различных программных продуктов существенно затрудняет возможность сбора, обработки и анализа основных метрик и показателей на всем протяжении бизнес-процесса.

Кроме вышеперечисленных, к выявленным недостаткам, существующей модели бизнес-процесса, следует отнести: отсутствие системы резервного копирования и восстановления данных, отсутствие возможности разграничения контроля и управления уровнем доступа сотрудников, отсутствие удобного и понятного пользовательского интерфейса, ограниченные возможности масштабирования в случае роста и развития предприятия.

### 1.8 Определение бизнес-требований

Учитывая вышеописанные недостатки и проведенный анализ бизнес-процесса AS-IS, а также основываясь на поставленных, перед предприятием, долгосрочных целях и задачах, сформирован следующий перечень предложений по его улучшению:

– сотрудники должны взаимодействовать друг с другом внутри единого информационного пространства с возможностью онлайн доступа;
– поданные заявки на оплату должны автоматически проверятся на соответствие действующей учетной политики;
– необходимо обеспечить возможность систематизации, хранения и поиска информации в рамках функционала бизнес-процесса;
– классификация заявок по статьям затрат должна выполняться автоматизировано, с привлечением сотрудника экономического отдела только для возможной проверки корректности;
– любые обработки и преобразования форматов, согласно утвержденному перечню документов, должны осуществляться в автоматизированном режиме;
– вся информация должна автоматически сохранятся в отдельном хранилище для предотвращения ее порчи злонамеренно или вследствие сбоя.

В соответствии со стратегией развития и планами по информатизации деятельности предприятия реализацию предложенных возможностей по улучшению следует провести посредством внедрения автоматизированной информационной системы. Внедрение АИС, соответствующей вышеописанным бизнес-требованиям, позволит предприятию сократить трудовые и временные затраты, повысить скорость работы, улучшить контроль и оперативное управление, перераспределить рабочую нагрузку, а также собрать, для дальнейшего анализа, ключевые показатели эффективности.

### 1.9 Анализ существующих решений

Для проведения анализа были отобраны следующие программные продукты: Directum RX, DocSpace и Битрикс24.

Directum RX разработка отечественного ИТ предприятия Directum, которое входит в ТОП-10 поставщиков корпоративных решений на территории РФ и стран СНГ по данным Tadviser. Компания имеет широкую партнерскую сеть, учебные заведения и множество успешных проектов внедрения. Программное решение Directum RX позволяет успешно выполнять проекты цифровизации различного масштаба и сложности включая такие задачи, как: автоматизации делопроизводства, работы кадровой службы, управления бизнес-процессами и другие. Кроме того, Directum RX имеет встроенные возможности интеллектуальной обработки первичной документации, писем, чеков и протоколов.

Система DocSpace разработана компанией Conteq на основе продукта Microsoft SharePoint 2013. Компания Conteq является как поставщиком готовых программных продуктов, так и разрабатывает под заказ IT-решения и корпоративные сервисы, а также выполняет весь комплекс работ по консалтингу и внедрению IT систем для среднего и крупного бизнеса. DocSpace содержит функционал по автоматизации и управлению электронным документооборотом, возможности цифрового архива с системой поиска, аналитический модуль для отслеживания показателей, а также имеет сертификацию ФСТЭК и соответствует ГОСТ по автоматизации делопроизводства.

Битрикс24 это известный продукт на отечественном рынке, который содержит большой набор модулей и компонентов для организации совместной работы включающий: CRM, видео звонки, управление проектами, онлайн офис для удаленной работы, интеграцию с мессенджерами и контакт центр. Компания разработчик работает на Российском рынке более 20 лет успешно занимаясь созданием корпоративных информационных систем. Располагает разветвленной сетью партнеров и большим числом специалистов для решений задач различного уровня.

Критерии и сравнительный анализ выбранных продуктов, на основании ранее сформулированных бизнес-требований, приведен в таблице 3.

Таблица 3 – Сравнительный анализ существующих решений

| Критерий | Directum RX | DocSpace | Битрикс24 |
|---|---|---|---|
| Заявки на оплату с настраиваемыми полями ввода | Да | Да | Да |
| Построение произвольной логики бизнес-процессов | Да | Да | Да |
| Реестры платежей по форме предприятия | Нет | Нет | Нет |
| Справочники для учета контрагентов, сотрудников, проектов, статей затрат, плательщиков | Да | Да | Да |
| Поиск по ключевым полям, дате и тексту | Да | Да | Да |
| Настраиваемые отчеты и выгрузки | Доработка | Доработка | Доработка |
| Автоматизированная классификация по статьям затрат | Нет | Нет | Нет |
| Self-hosted размещение | Да | Да | Да |
| Интеграция с Active Directory, PowerBI и 1С Предприятием | Да | Да | Маркет |
| Цена лицензирования на 100 пользователей | от 1 037 000 руб. | от 730 000 руб. | 199 000 руб. |
| Цена внедрения в инфраструктуре предприятия | от 370 000 руб. | от 230 000 руб. | от 150 000 руб. |
| Стоимость дальнейшего владения и сопровождения | Высокая | Высокая | Средняя |

На основании сведений, приведенных в таблице 4, можно заключить, что все решения обеспечивают большую часть предъявляемых бизнес-требований, но для полного покрытия поставленных задач потребуется выполнить дополнительную разработку, реализовав реестр платежей по форме предприятия и автоматизированную классификацию по статьям затрат. Кроме того, рассмотренные решения предоставляют гораздо более обширный функционал, что отражается на итоговой стоимости лицензий и внедрения, а также требует привлечения квалифицированных специалистов, прошедших обучение по работе с системами. Учитывая, что руководством предприятия

поставлена задача автоматизации только бизнес-процесса управления платежами, а также стремление минимизировать расходы, то разработка и внедрение собственной системы является рационально обоснованным решением.

### 1.10 Требования к разрабатываемой АИС

На данном этапе необходимо определить цели, формализовать основные требования и задачи к разрабатываемой АИС. Сами по себе требования должны соответствовать общепризнанными критериями качества [11]. Некоторые авторы используют в своих работах простое текстовое описание с разделением на функциональные и нефункциональные требования. Также известно, что в проектах информатизации может применяться классификация FURPS, подход Вигерса и модель, предложенная в PMBOK [10]. В настоящей работе применяется модель FURPS, поскольку, по мнению автора, она позволяет на достаточном уровне описать предъявляемые требования к разрабатываемой АИС.

АИС «Payman» разрабатывается по заданию предприятия с целью автоматизации бизнес-процесса управления платежами для устранения выявленных недостатков и реализации определенных ранее бизнес-требований. Соответствие бизнес-требованиям осуществляется за счет автоматизации основных этапов бизнес-процесса, снижения нагрузки сотрудников путем исключения рутинных операций, улучшения пользовательского опыта и внедрения алгоритмов машинного обучения.

Ответственным за реализацию проекта по разработке и внедрению системы назначается руководитель экономического отдела. Заинтересованными лицами и конечными пользователями АИС являются все сотрудники предприятия, которые принимают непосредственное участие в бизнес-процессе управления платежами, согласно внутренним регламентам предприятия.

Функциональные требования к системе:

- должна предусматривать авторизацию с помощью логина и пароля, журналирование входа и возможность управления пользователями;
- обеспечивать ведение (создание, изменение и удаление записей) списков с полями, указанными в Приложении А;
- списки заявок и реестров должны поддерживать постраничный вывод, сортировку и возможность фильтрации;
- реестр заявок должен выполнять автоматический расчет итогов;
- изменение статуса реестра заявок должно сопровождаться автоматическим уведомлением по e-mail;
- классификация по статьям затрат поданной заявки на оплату должна выполняться автоматически согласно учетной политики с возможностью последующего изменения в ручном режиме;
- должна обеспечивать выгрузку утвержденного реестра платежей в Microsoft Excel согласно формату, который показан в Приложении Б.

Требования по удобству использования системы:

- экранные формы должны обеспечивать удобочитаемость, высокую скорость загрузки и корректно отображаться на мониторе с разрешением 1600 на 1024 точек и выше;
- визуальная компоновка не должна нарушаться при изменении размера окна браузера, системного шрифта или других настроек;
- оформление должно быть выполнено в едином стиле с комфортным цветом фона и размером шрифта, а также иметь понятную навигацию и организацию экранных форм;
- документация должна включать руководство пользователя.

Требования к надежности системы:

- должна поддерживать резервное копирование и восстановление;
- должна обеспечивать возможность сохранения состояния при кратковременных сбоях или потерях соединения;

– при вводе ошибочных данных или непреднамеренных действий пользователя сохранять корректную работоспособность;
– предусматривать работу в круглосуточном режиме, а общее время простоя системы по причине ее неработоспособности или проведения восстановительных работ не должно превышать 2 часов в месяц.

Требования к производительности системы:

– поддерживать одновременную работу до 100 пользователей;
– обеспечивать обработку и хранение данных объемом до 100 ГБ;
– обеспечивать полнофункциональную работу на выделенном сервере с двумя процессорными ядрами и 8 ГБ оперативной памяти.

Требование к поддерживаемости:

– позволять выполнение регламентного обслуживания системы сотрудниками предприятия без привлечения внешних специалистов;
– предусматривать возможность дальнейшего развития функционала и увеличения количества одновременно работающих пользователей.

Кроме вышеперечисленных требований, при разработке системы необходимо применять реляционную модель организации базы данных, используемые инструменты и компоненты должны распространяться по бесплатной модели лицензирования, выбранный язык программирования должен поддерживать объектно-ориентированный подход.

### 1.11 Модель бизнес-процесса TO-BE

После проведенного анализа и формирования требований к будущей системе следует выполнить разработку модели бизнес-процесса TO-BE. Если АИС будет опираться на существующую модель AS-IS, то это приведет к закреплению существенных недостатков и не позволит в полной мере реализовать возможности улучшения, роста и развития предприятия [8]. Контекстная модель представлена на рисунке 5.

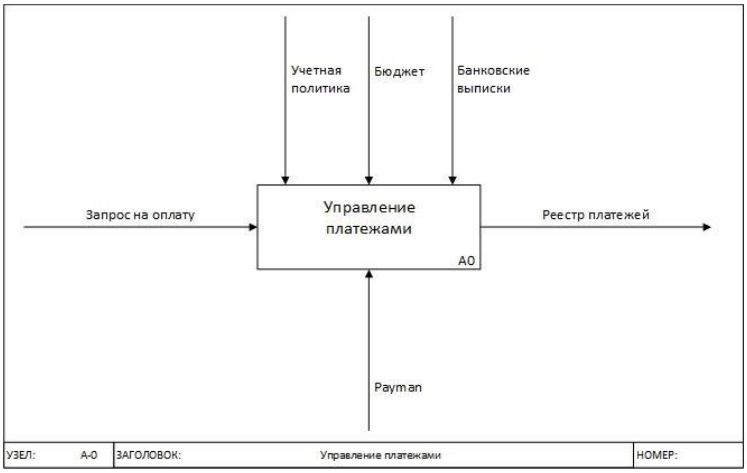


Рисунок 5 – Контекстная диаграмма модели процесса TO-BE

Модель процесса TO-BE, показанная на рисунке 5, сохранила интерфейсные дуги модели AS-IS по управлению и выходу. Однако, формирование заявок будет проводится в системе, а в качестве механизмов вместо E-mail, Microsoft Excel и Google Sheets используется АИС «Payman». Диаграмма декомпозиции показана на рисунке 6.

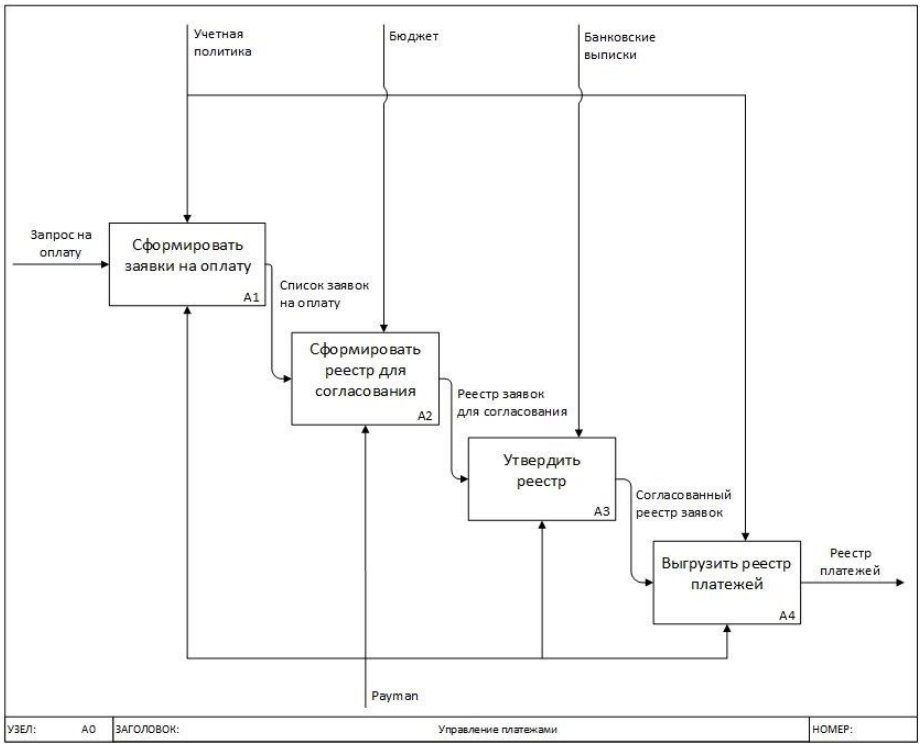


Рисунок 6 – Диаграмма декомпозиции модели процесса TO-BE

Согласно модели TO-BE, сотрудник, используя выданный логин и пароль, должен пройти авторизацию в АИС и сформировать заявку на оплату в личном кабинете. Каждая новая заявка заполняется в соответствии с учетной политикой, что обеспечивается набором встроенных справочников АИС. При занесении новых заявок система автоматически выполняет классификацию и присвоение соответствующих статьей затрат каждой заявке. Начальник экономической службы, основываясь на БДДС и БДР, выполняет формирование реестра заявок для согласования и передает его на утверждение генеральному директору. Генеральный директор, как и ранее, может одобрить, отклонить, уменьшить сумму выплаты или полностью отложить платеж по любой заявке из реестра. Согласованный реестр заявок при помощи АИС выгружается в реестр платежей, который передается в бухгалтерию для последующей оплаты. Для анализа информационных потоков данных удобно применить методологию DFD. Диаграмма декомпозиции потоков данных модели TO-BE показана на рисунке 7.

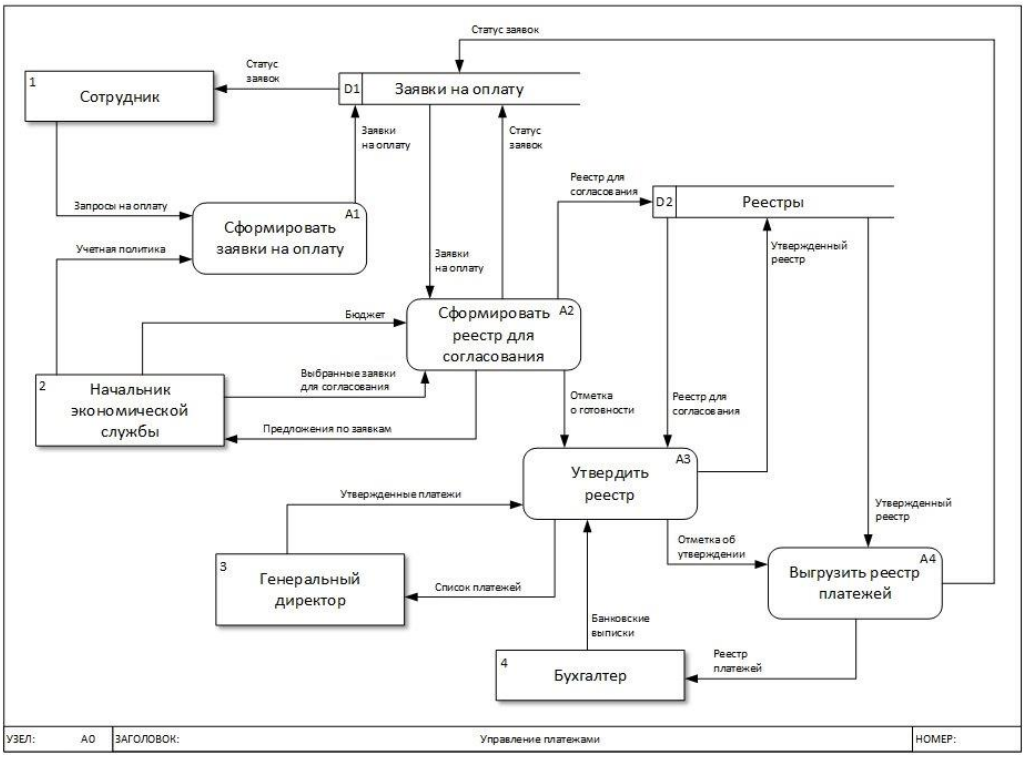


Рисунок 7 – Диаграмма декомпозиции потоков данных модели TO-BE

Диаграмма, приведенная на рисунке 7, позволяет определить основные потоки данных, внешние сущности, хранилища данных и процессы. Представленные в настоящей главе сведения о модели бизнес-процесса управления платежами позволят провести проектирование АИС.

## Выводы по первой главе

Устойчивый и быстрый рост компании Проектное бюро R1 привел к запросу на модернизацию протекающих бизнес-процессов. Одним из бизнес-процессов верхнего уровня является процесс управления финансами, который контролируется финансово-юридическим отделом под общим руководством финансового директора. Проведя сравнительный анализ и учитывая масштаб, особенности и различные критерии отбора, среди различных применяемых методологий анализа и моделирования предметной области выбраны семейство методологий IDEF и DFD. Анализ модели бизнес-процесса управления платежами AS-IS позволил выявить ряд существенных недостатков, в частности: использование разнородных систем и средств, дублирование одинаковых данных, отсутствие единой информационной среды, а также другие. Указанные недостатки влекут за собой увеличение ошибок, неточностей, трудовых и временных затрат, а также снижение скорости обработки и управленческой эффективности. Автоматизация и внедрение АИС позволит предприятию убрать вышеописанные недостатки. Проведенное исследование представленных программных решений показало, что на рынке присутствуют системы способные реализовать предъявляемые требования только частично. Из чего следует, что разработка и внедрение собственной АИС является рационально обоснованным решением. Для постановки задачи и определения требований к АИС применялась модель классификации FURPS. Построенная модель бизнес-процесса управления платежами TO-BE позволит реализовать сформулированные требования и исправить выявленные недостатки.

# Глава 2 Проектирование АИС «Payman»

## 2.1 Выбор технологии проектирования

После этапа анализа и моделирования необходимо перейти к проектированию и рассмотреть особенности разработки АИС. Ранее отмечалось, что существуют различные методологии и нотации, которые успешно применяются для разработки и проектирования. Поскольку в сформулированных требованиях к АИС присутствует указание на применение языков программирования, которые поддерживают ООП, то целесообразно рассмотреть возможность применения комплементарной методологии.

Унифицированный язык моделирования UML позволяет проводить работы по проектированию и документированию на протяжении всего цикла разработки, обеспечивая возможность построения графиков и диаграмм. В отличие от подходов, которые основаны на функциональной декомпозиции анализируемого процесса, в объектно-ориентированном подходе система рассматривается как множество классов и объектов, обладающих определенными свойствами и поведением [17]. Среди прочих преимуществ такого подхода стоит отметить возможность применения таких концепций, как: инкапсуляция, наследование, полиморфизм, модульность и абстракция. Все это повышает продуктивность разработки, сокращает затрачиваемое время и улучшает общее качество проектных решений [22]. Заложенные принципы и особенности позволяют UML оставаться одним из наиболее перспективных подходов в проектировании и разработке ИС [12].

Основываясь на собранной информации, можно заключить, что применение UML в процессе проектирования АИС является разумным и обоснованным решением.

Диаграммы UML можно разделить на две группы: статические и динамические. Статические диаграммы определяют константные сущности и их взаимосвязи. К статическим относятся диаграммы классов, объектов,

компонентов и развертывания. На динамических диаграммах представляют протекающие процессы. К динамическим относятся диаграммы деятельности, последовательности, взаимодействия и состояний.

## 2.2 Варианты использования

Диаграмма вариантов использования применяется для представления внешних, по отношению к системе, пользователей (акторов), а также доступный функционал, который реализует разрабатываемая система и их связи [5]. Действующие лица иллюстрируются в виде силуэтов человечков, а функционал системы рассматривается как множество прецедентов и изображается эллипсами. Диаграмма вариантов использования показана на рисунке 8.

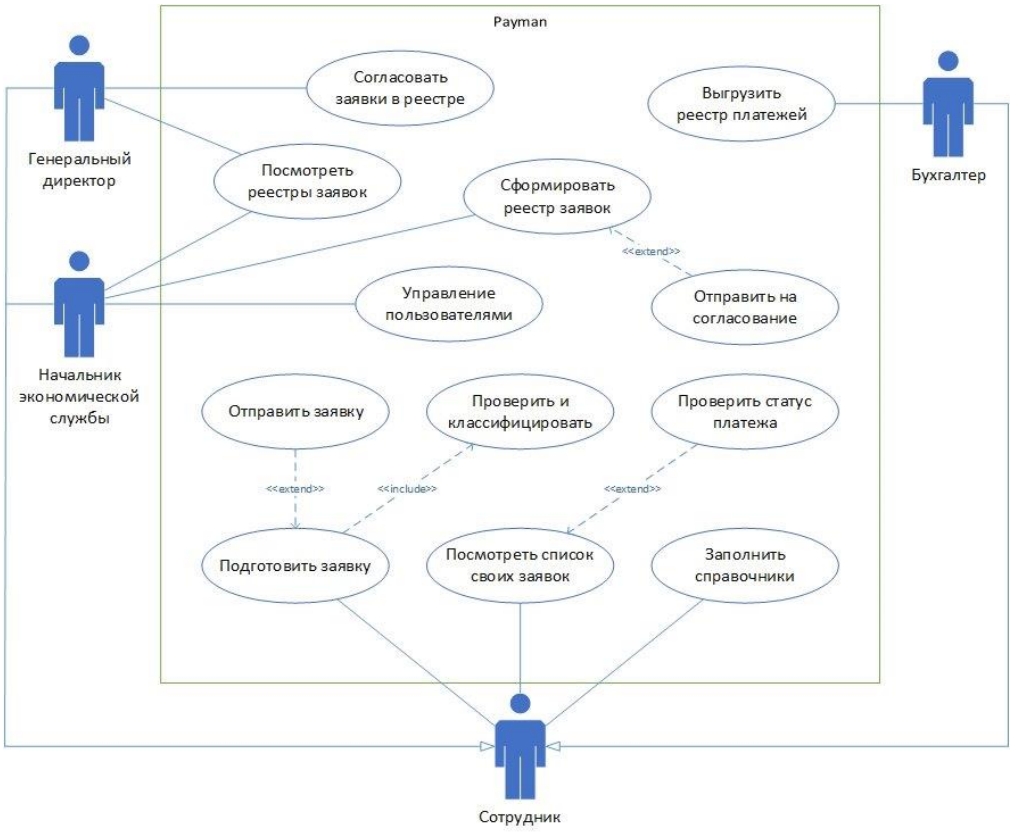


Рисунок 8 – Диаграмма вариантов использования

Для уточнения подробностей взаимодействия, в рамках вариантов использования, применяются сценарии прецедентов. Описание прецедента «Подготовить заявку» приведено в таблице 4.

Таблица 4 – Описание прецедента «Подготовить заявку»

| Вариант использования «Подготовить заявку» | |
|---|---|
| Краткое описание | позволяет сотруднику предприятия создавать или изменять существующую заявку на оплату |
| Действующие лица | сотрудник |
| Предусловия | необходима авторизация в системе по предоставленному логину и паролю, существуют требуемые данные в справочниках системы |
| Основной поток | – начинается при нажатии кнопки «добавить» или «изменить» действующего лица в личном кабинете;<br>– система отображает форму заявки на оплату с полями для ввода данных описанным в Приложении А;<br>– сотрудник подготавливает заявку путем заполнения требуемых полей;<br>– сотрудник нажимает кнопку «сохранить»;<br>– система проверяет заполненные поля;<br>– если поля заполнены корректно, то система выполняет классификацию заявки согласно учетной политики по статье затрат;<br>– система выводит сообщение об успешной подготовке заявки;<br>– вариант использования завершается. |
| Альтернативный поток А1: Ошибки в заполнении полей ввода | – система сохраняет введённые действующим лицом значения в полях ввода данных;<br>– система информирует сотрудника о допущенных ошибках;<br>– действующее лицо возвращается к вводу данных;<br>– вариант использования завершается. |
| Постусловия | если заявка на оплату подготовлена успешно, то сохраняется в базе данных и становится доступна в личном кабинете действующего лица, в обратном случае система возвращается в начальное состояние |

Описание прецедента «Посмотреть список своих заявок» приведено в таблице 5.

Таблица 5 – Описание прецедента «Посмотреть список своих заявок»

| Вариант использования «Просмотреть список своих заявок» | |
|---|---|
| Краткое описание | позволяет сотруднику предприятия просматривать имеющиеся в системе свои заявку на оплату |
| Действующие лица | сотрудник |
| Предусловия | необходима авторизация в системе по логину и паролю |
| Основной поток | – начинается при переходе актора в личный кабинет;<br>– система отображает список имеющихся заявок;<br>– вариант использования завершается. |
| Альтернативный поток А1: Удаление заявки | – сотрудник выбирает заявку для удаления;<br>– если заявка не отправлена на согласование, то система удаляет заявку и выводит подтверждение;<br>– вариант использования завершается. |

Описание прецедента «Отправить заявку» приведено в таблице 6.

Таблица 6 – Описание прецедента «Отправить заявку»

| Вариант использования «Отправить заявку» | |
|---|---|
| Краткое описание | позволяет передать заявку на оплату для последующего добавления в реестр согласования |
| Действующие лица | сотрудник |
| Предусловия | необходима авторизация в системе, заявка подготовлена |
| Основной поток | – начинается при переходе актора в выбранную заявку;<br>– актор проверяет данные в заявке;<br>– если все заполненные данные в полях ввода соответствуют требуемым значениям, то актор нажимает кнопку «отправить»;<br>– если какие-то данные в полях не соответствуют требуемым, то сотрудник вносит изменения;<br>– если были внесены изменения в заявку, то система выполняет проверку и классификацию;<br>– система изменяет статус заявки;<br>– система оповещает сотрудника об успешной отправке заявки на оплату;<br>– вариант использования завершается. |
| Постусловия | отправленная заявка на оплату лишается возможности дальнейшего изменения |

Описание прецедента «Сформировать реестр заявок» дано в таблице 7.

Таблица 7 – Описание прецедента «Сформировать реестр заявок»

| Вариант использования «Сформировать реестр заявок» | |
|---|---|
| Краткое описание | позволяет формировать реестры на согласование, которые включают ранее поданные заявки на оплату |
| Действующие лица | начальник экономической службы |
| Предусловия | необходима авторизация в системе, существуют подготовленные заявки, заполнены справочники плательщиков и данные банковских счетов плательщиков |
| Основной поток | – начинается при переходе в модуль реестры;<br>– актор нажимает кнопку «добавить»;<br>– система показывает форму, в которой актор указывает название и отмечает плательщиков;<br>– актор выбирает плательщика;<br>– при нажатии кнопки «добавить заявки» система отображает список доступных к добавлению заявок;<br>– актор отмечает необходимые заявки;<br>– система рассчитывает сводные итоги по реестру;<br>– актор нажимает кнопку «сохранить»;<br>– система оповещает об успешном сохранении;<br>– вариант использования завершается. |
| Альтернативный поток А1: Удаление заявок в реестре | – если реестр не отправлен на согласование актор переходит в режим редактирования;<br>– актор удаляет выбранные заявки;<br>– система оповещает об успешном удалении;<br>– вариант использования завершается. |
| Альтернативный поток А2: Изменение суммы платежа в заявке на оплату | – если реестр не отправлен на согласование, то переходит в режим редактирования реестра;<br>– актор в строке требуемой заявки на оплату проставляет новую сумму платежа не превышающей исходного значения;<br>– система рассчитывает сводные итоги по реестру;<br>– актор нажимает кнопку «сохранить»;<br>– система оповещает об успешном сохранении;<br>– вариант использования завершается. |
| Альтернативный поток А4: Внесение остатков по банковским счетам | – если реестр не отправлен на согласование, то переходит в режим редактирования;<br>– актор, выбрав плательщика, вносит данные по остаткам на банковских счетах;<br>– актор нажимает кнопку «сохранить»;<br>– система сохраняет изменения и показывает оповещение;<br>– вариант использования завершается. |

Описание прецедента «Согласовать заявки в реестре» дано в таблице 8.

Таблица 8 – Описание прецедента «Согласовать заявки в реестре»

| Вариант использования «Согласовать заявки в реестре» | |
|---|---|
| Краткое описание | позволяет утверждать отправленные на согласование реестры |
| Действующие лица | генеральный директор |
| Предусловия | необходима авторизация в системе, реестр отправлен на согласование |
| Основной поток | – начинается при переходе в реестр заявок;<br>– актор просматривает каждого плательщика, выбранные заявки, суммы платежей и сводные итоги по реестру;<br>– актор может произвести корректировку суммы платежа по любой заявке на оплате в реестре;<br>– после корректировки суммы платежа система автоматически рассчитывает сводные итоги;<br>– актор для каждой заявки в реестр может выбрать «оплатить», «отклонить» или «частично оплатить»;<br>– после получения необходимых параметров актор нажимает кнопку «согласовать»;<br>– система изменяет статус реестра и выводит оповещение для актора;<br>– вариант использования завершается. |
| Альтернативный поток А1: Вернуть обратно на формирование | – при просмотре реестра актор может обнаружить неприемлемые данные и нажать кнопку «вернуть в работу»;<br>– система изменяет статус реестра и выводит оповещение;<br>– система оповещает начальника экономической службы о возврате реестра заявок в работу;<br>– вариант использования завершается. |
| Постусловия | – согласованный реестр заявок не подлежит дальнейшему удалению или изменению;<br>– если реестр заявок был согласован, то система должна произвести автоматическую выгрузку реестра платежей. |

Для вариантов использования «Посмотреть реестры заявок», «Управление пользователями», «Заполнить справочники», «Выгрузить реестр платежей» и «Авторизация» не будет предоставлено описание, так как подразумевается, что взаимодействие с системой осуществляется согласно общепринятым для этого функционала способом.

## 2.3 Диаграмма классов

Основываясь на полученной концептуальной модели предметной области можно построить диаграмму классов. Диаграмма классов включает информацию о классах предметной области и их взаимосвязях [21]. Класс олицетворяет сущность, которая содержит данные и обладает поведением. На диаграмме классы обозначаются прямоугольником, в котором задано имя класса, перечень его свойств и операций. При построении диаграммы применялись следующе типы связей: ассоциация, генерализация и композиция. Связь по ассоциации показывает взаимодействие двух классов. Связь по генерализации означает, что дочерний класс наследует свойства, состояние и поведение родительского класса. Связь по композиции представляет отношения вида целое-часть с тем условием, что часть не может существовать отдельно от целого. Диаграмма классов показана на рисунке 9.

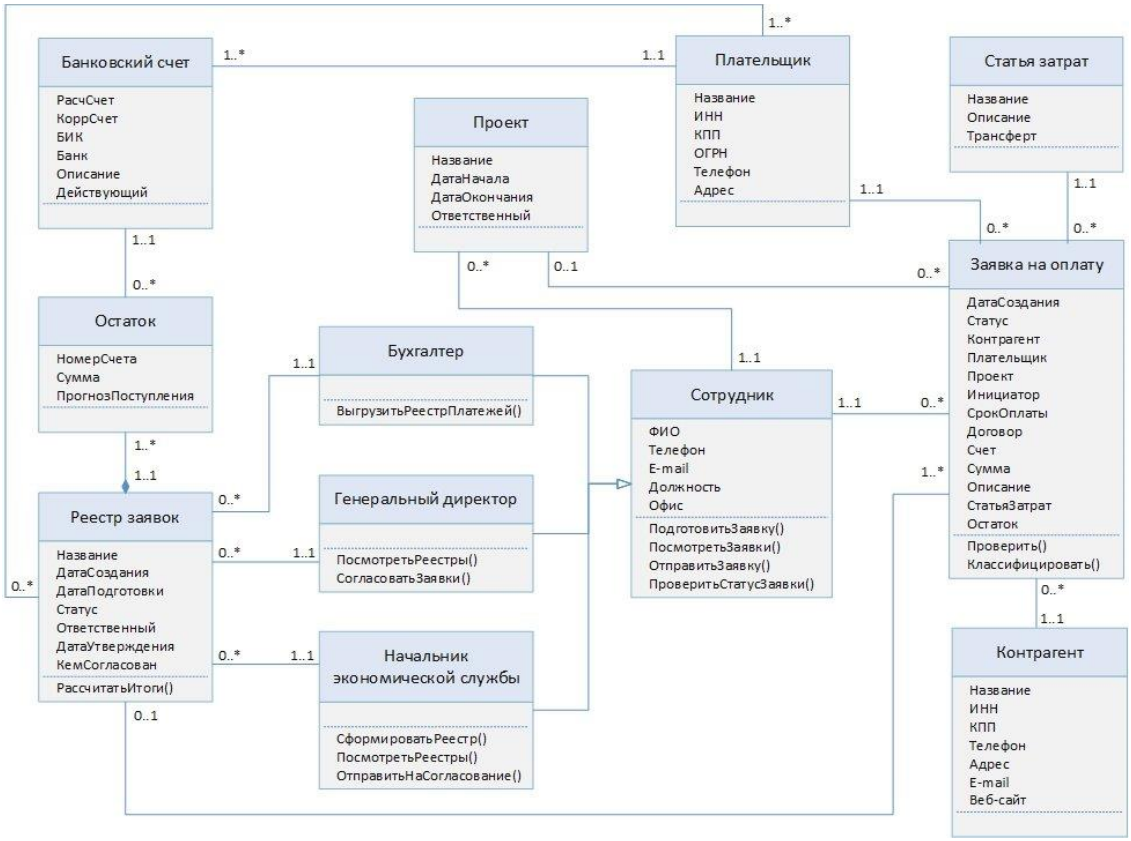


Рисунок 9 – Диаграмма классов бизнес-процесса управления платежами

На диаграмме классов бизнес-процесса, показанной на рисунке 9, классы «Бухгалтер», «Генеральный директор», «Начальник экономической службы» являются обобщением класса «Сотрудник» потому, что эти сущности должны наследовать все свойства, связи и поведение родительского класса. Класс «Сотрудник» связан отношением ассоциации с классами «Заявка на оплату» и «Проект», поскольку необходимо обеспечить двунаправленную навигацию между этими классами. Класс «Контрагент» связан с классом «Заявка на оплату» отношением ассоциации так как для каждой заявки на оплату должен быть указан контрагент, в отношении которого выполняется оплата. Поскольку, каждая заявка на оплату, согласно учетной политике, должна быть классифицирована статьей затрат, то класс «Заявка на оплату» имеет ассоциацию с классом «Статья затрат». Заявка на оплату должна выполняться определенным плательщиком, что означает необходимое наличие ассоциации между классами «Заявка на оплату» и «Плательщик». Поскольку некоторые заявки на оплату могут относиться к определенным проектам, то между классами «Заявка на оплату» и «Проект» установлена отношение ассоциации. Заявки на оплату добавляются в реестры на оплату, что влечет необходимое наличие связи ассоциации между классами этих сущностей. Поскольку при формировании реестра на оплату, начальник экономического отдела заносит данные по текущим остаткам и прогнозам поступлений на банковские счета, которые являются неотъемлемой частью реестра, то классы связаны отношением композиции. В свою очередь классы «Бухгалтер», «Генеральный директор», «Начальник экономического отдела» в разной степени взаимодействуют с реестром заявок, что приводит к связи ассоциации.

### 2.4 Диаграмма деятельности оплаты заявок

Диаграмма деятельности предоставляет возможность показать потоки управления и множество действий из которых состоит рассматриваемая активность. Допускается комбинировать различные варианты использования

для лучшего представления процесса в целом [19]. На рисунке 10 представлена диаграмма деятельности оплаты заявки.

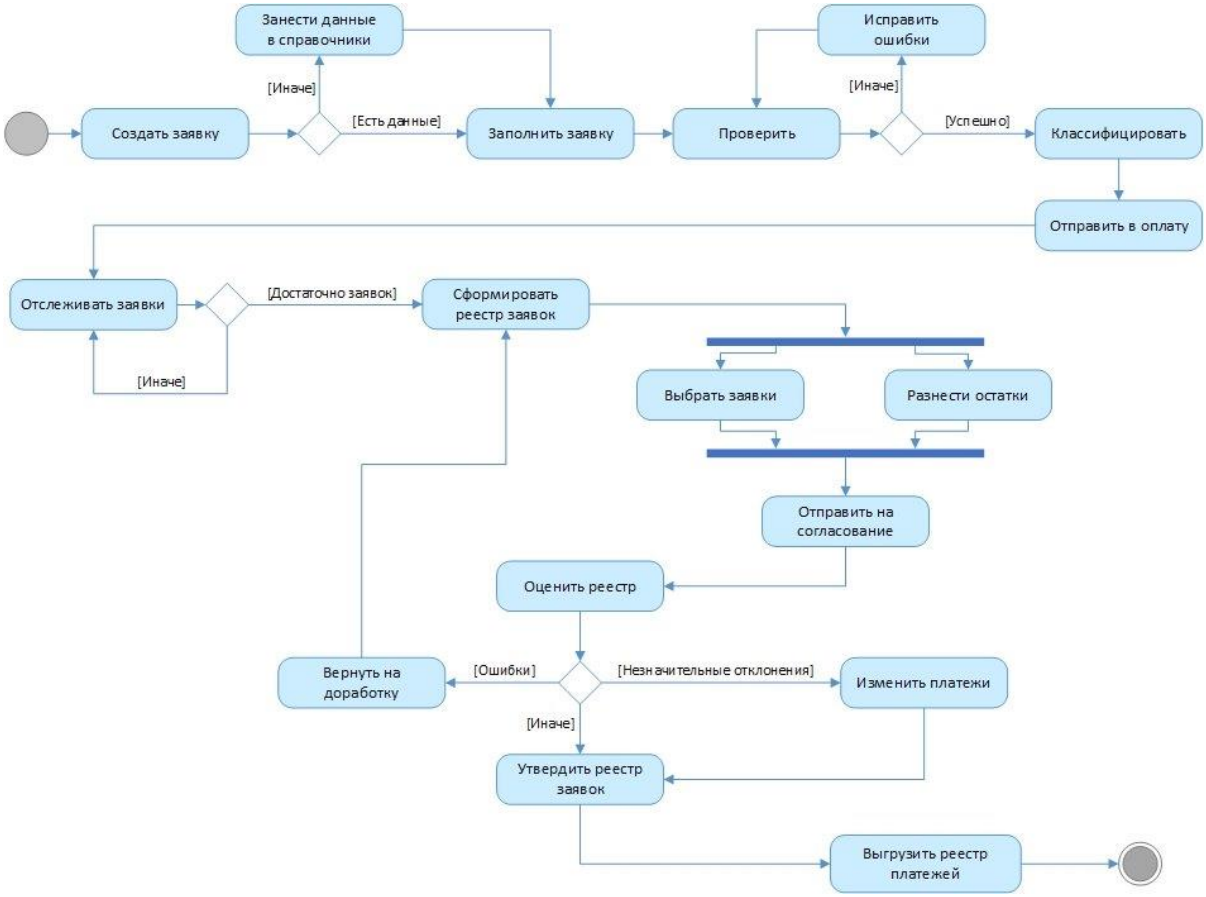


Рисунок 10 – Диаграмма деятельности оплаты заявок

Построение диаграммы деятельности, как наглядно демонстрирует рисунок 10, позволяет существенно улучшить понимание всего процесса оплаты по заявкам в целом. С более обстоятельным описанием этого процесса можно ознакомиться в предшествующих параграфах настоящей работы.

### 2.5 Диаграммы состояний

Заявки и реестры на оплату, в зависимости от текущей логической стадии процесса, могут находится в различных состояниях. Каждое состояние, в котором находится сущность, обладает определенным поведением и

свойствами. Для демонстрации возможных состояний и переходов между ними может применяться диаграмма состояний [23]. Диаграмма состояний заявки на оплату представлена на рисунке 11.

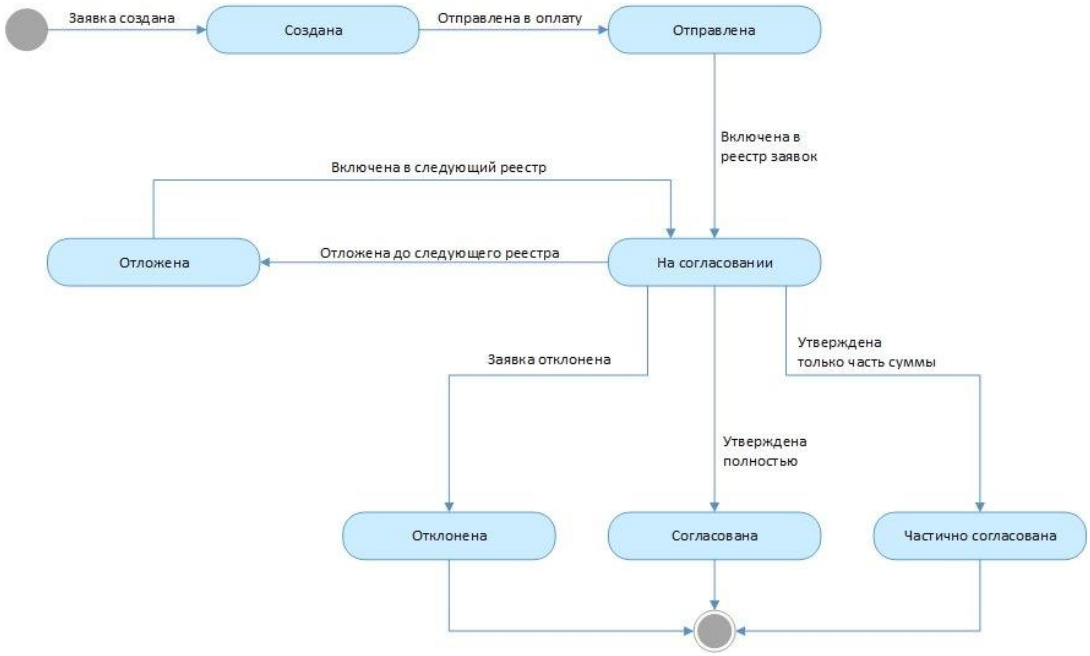


Рисунок 11 – Диаграмма состояний заявки на оплату

После создания, как видно из рисунка 11, заявка на оплату переходит в состояние «создана». В этом состоянии возможно изменить внесенные данные и проверить правильность заполнения полей. Когда все данные заполнены и проверены необходимо отправить заявку в оплату. После этого состояние переходит в «отправлена». В этом и последующих состояниях любые изменения или уточнения ранее внесенных данных запрещены. Переход в состояние «отправлена» позволяет начальнику экономической службы включать эту заявку в формируемый реестр для дальнейшего согласования. После отправки реестра, который содержит данную заявку, на согласование генеральному директору заявка переходит в состояние «на согласовании». В этом состоянии заявка будет находится до тех пор, пока генеральный директор принимает решение о возможности и целесообразности оплаты. В результате его решения заявка может быть отклонена, согласована, частично согласована

или отложена. Частичное согласование подразумевает уменьшение запрошенной суммы платежа в заявке на оплату. Если было принято решение отложить оплату по заявке, то тогда заявка переходит в состояние «отложена» и становится доступной для включения в последующие реестры.

Для реестра заявок на оплату составлена диаграмма состояний, которая показана на рисунке 12.


Рисунок 12 – Диаграмма состояний реестра заявок на оплату

Как можно заметить из диаграммы, показанной на рисунке 12, множество переходов состояний для реестра на оплату выглядит довольно прямолинейно. Однако стоит отметить, что реестр на оплату при возврате на доработку генеральным директором, переходит в состояние «создан» и не имеет отдельного состояния для этого случая.

### 2.6 Диаграмма компонентов

Для демонстрации логической структурной организации архитектуры АИС можно построить диаграмму компонентов. Компонент является некоторой структурной сущностью, которая скрывает свою внутреннюю реализацию определенного функционала и обладает внешними интерфейсами взаимодействия. Представление системы с помощью диаграммы компонентов позволяет обозначить границы функциональности, которая инкапсулируются внутри компонента и может быть заменена другими компонентами при реализации соответствующих интерфейсов.

В приведенной ниже диаграмме, кроме компонентов, используются интерфейсы и порты. Интерфейс задает заранее определенное и фиксированное множество функций, которые реализует или запрашивает компонент. Порт – это конкретная точка взаимодействия компонента с внешней средой, которая обеспечивает инкапсуляцию компонента.

Если компонент делегирует реализацию внешнего интерфейса некоторой конкретной внутренней сущности, то на диаграмме необходимо указывать порт через которой проходят входящие или исходящие сообщения. Диаграмма компонентов представлена на рисунке 13.

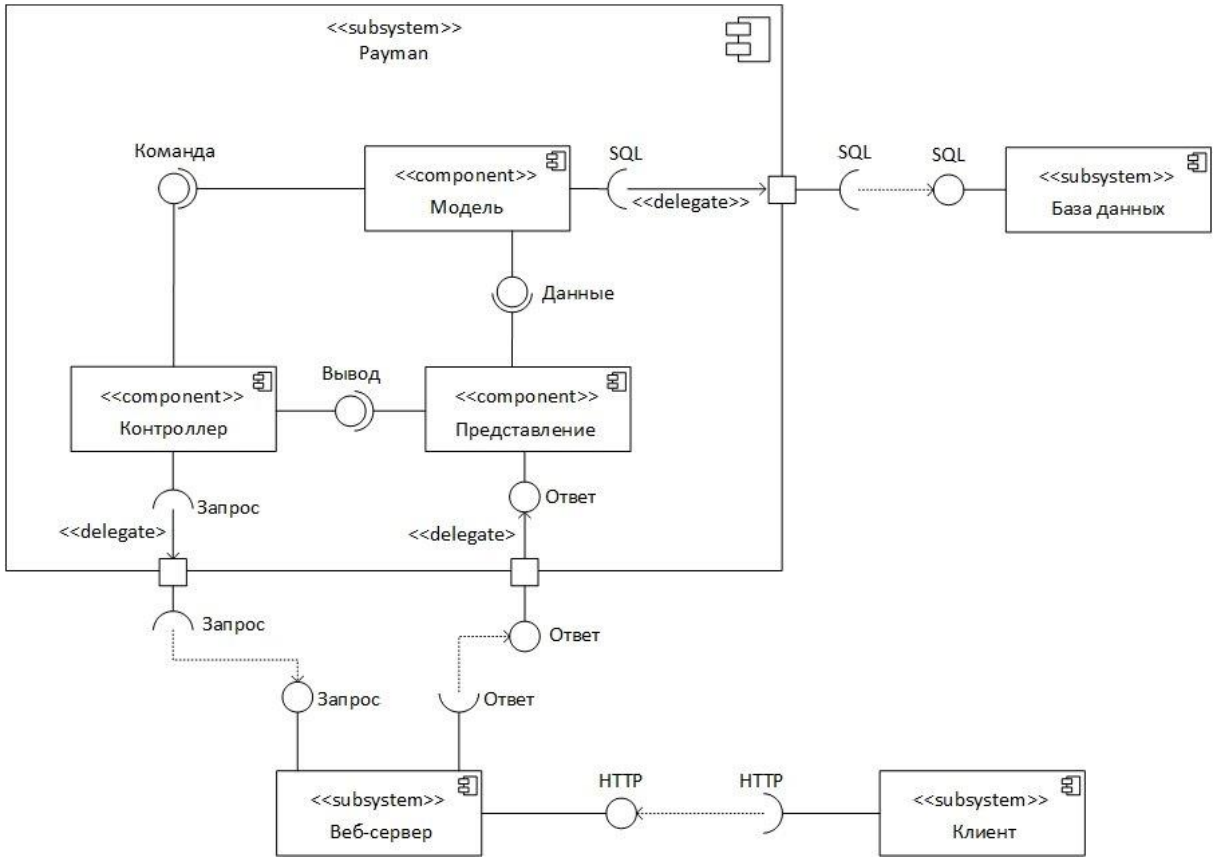


Рисунок 13 – Диаграмма компонентов

Архитектура системы представлена в виде четырех подсистем. Подсистема «Payman» реализует архитектуру и концепцию MVC. Паттерн MVC позволяет обеспечить разделение ответственности внутри

информационной системы на три логические компоненты: управление данными, управление логикой и управления отображением.

Как видно из рисунка 13, подсистема «Payman» имеет порты для делегирования внешних интерфейсов и состоит из трех компонентов: «Контроллер», «Модель» и «Представление».

Компонент «Контроллер» реализует логику обработки поступающих запросов путем управления моделью с помощью команд и вывода соответствующего представления. Взаимодействие этого компонента обеспечивается через предоставляемые интерфейсы «Команда» и «Вывод», а требует для своей работы интерфейс «Запрос».

Компонент «Модель» реализует логику управления данными. По команде от контроллера, через требуемый интерфейс «Команда», взаимодействуя с базой данных, через требуемый интерфейс «SQL», передает данные для компонента «Представление», через предоставляемый интерфейс «Данные».

Компонент «Представление» обеспечивает логику формирования ответа получая данные, через требуемый интерфейс «Данные», руководствуясь инструкциями по выводу, через требуемый интерфейс «Вывод», формирует и передает ответ через предоставляемый интерфейс «Ответ».

Подсистема «База данных» обеспечивает хранение и обработку поступающих запросов через предоставляемый интерфейс «SQL».

Подсистема «Веб-сервер» реализует логику управления и взаимодействия с клиентами через предоставляемый интерфейс «HTTP». Для передачи поступающих от клиентов запросов имеет предоставляемый интерфейс «Запрос» и требуемый интерфейс «Ответ» для получения результирующего вывода.

Подсистема «Клиент» реализует логику работы с конечным пользователем и имеет требуемый интерфейс «HTTP».

## 2.7 Диаграмма развертывания

Для демонстрации структуры размещения системы может применяться диаграмма развертывания. Узлом называется отдельное физическое или виртуальное вычислительное устройство, на котором размещаются компоненты системы. На диаграмме узел изображается в виде отдельно стоящего куба. Внутри узла могут быть расположены компоненты системы. Взаимосвязи между узлами обозначаются прямыми линиями без стрелок. Каждая взаимосвязь может быть снабжена показателем мощности связи. Кроме узлов, компонентов и связей на диаграмме развертывания могут присутствовать артефакты. Артефакт — это некоторый набор физической информации (файлы, сценарии, исполняемые модули). Диаграмма развертывания системы представлена на рисунке 14.

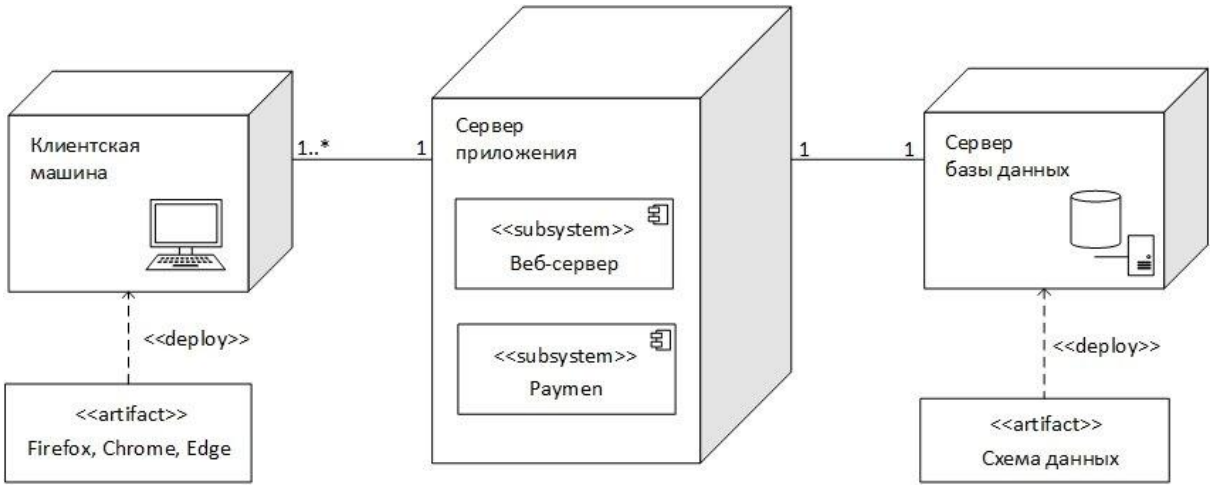


Рисунок 14 – Диаграмма развертывания

Система размещается на трех узлах. На узле клиентской машины устанавливается любой современный браузер, например, Firefox, Chrome или Edge. На узле сервера базы данных импортируется схема базы данных системы.

## 2.8 Концептуальная модель данных

Поскольку проектируемая система хранит, обрабатывает и обеспечивает доступ к данным, то одной из важных задач является разработка концептуальной модели данных. Концептуальная модель данных относится к информационным моделям предметной области и описывает сущности, атрибуты и их взаимосвязи. При проектировании модель рассматривается без привязки к конкретной технологии организации и хранения данных. В данной работе для разработки ER-диаграммы применялась графическая нотация П. Чена. Сущности на диаграмме изображаются с помощью прямоугольников, атрибуты с помощью овалов, а их связи с помощью ромбов. На рисунке 15 представлена ER-диаграмма концептуальной модели данных.

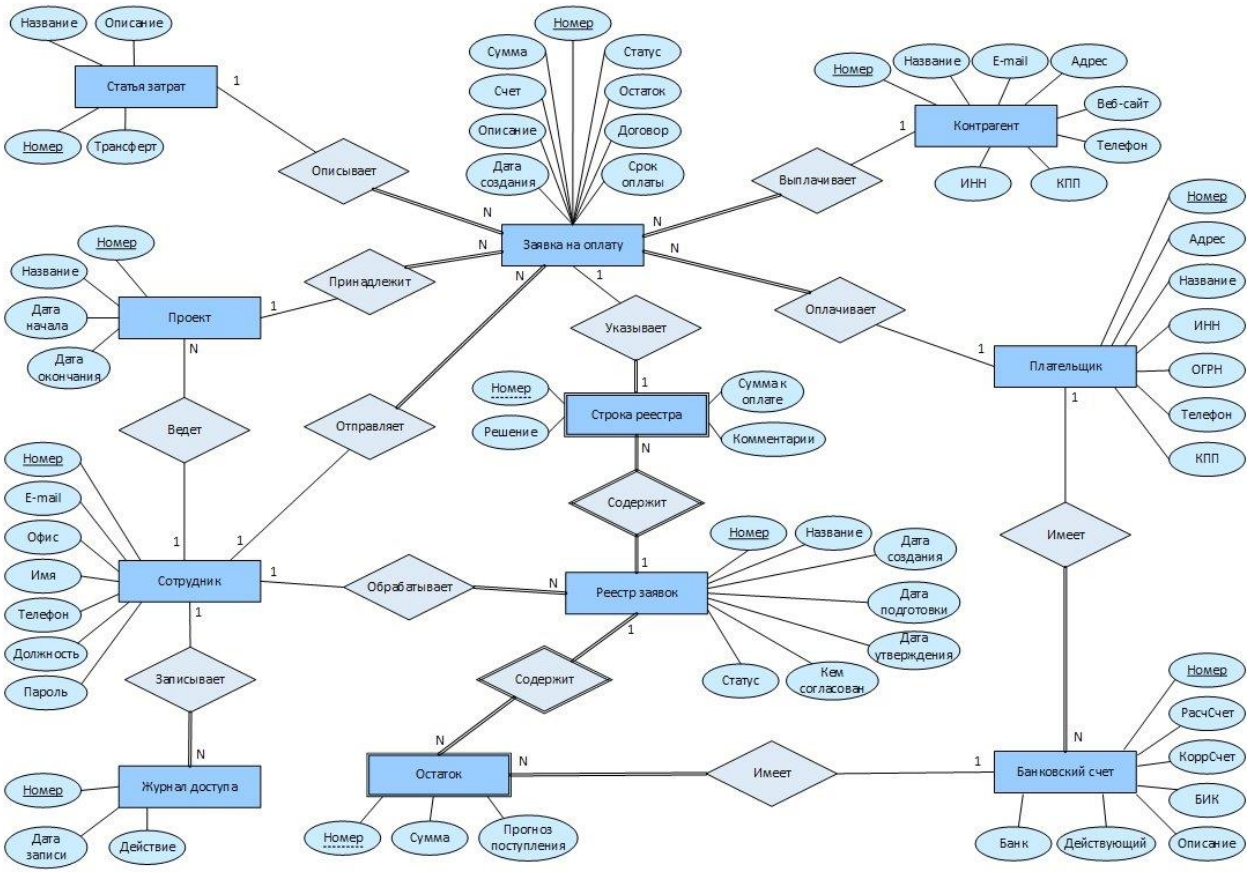


Рисунок 15 – ER-диаграмма концептуальной модели данных

Концептуальная модель данных, представленная на рисунке 15, содержит все необходимые данные для работы проектируемой системы.

Поскольку реестр заявок состоит из строк, где каждая строка указывает на конкретную заявку на оплату, а также по каждому плательщику указывается денежный остаток, то сущности «Остаток» и «Строка реестра» обозначены слабыми сущностям. Связь между этими слабыми сущностями и их родительскими сущностями обозначена двойной контурной линей.

Двойная контурная линия используется поскольку первичные ключи слабой сущности состоят из внешнего ключа родительской сущности и ее частичного ключа (подчеркнутый пунктирной линией атрибут).

Между сущностями «Заявка на оплату», «Статья затрат», «Плательщик», «Сотрудник» и «Проект» установлена обязательная связь. Это означает, что все заявки на оплату в системе должны иметь ссылку на обязательные сущности.

Также обязательная связь установлена между сущностями «Реестр заявок», «Остаток», «Строка реестра», а также «Остаток» должен ссылаться на «Банковский счет».

### 2.9 Логическая модель данных

Основываясь на проведенном анализе, сформулированных требованиях и учитывая результаты работы по проектированию, рационально и обоснованно, в качестве технологии организации базы данных выбрать реляционную модель.

Для целей моделирования будет применяться методология IDEF1X потому, что предоставляемые этой методологией средства и концепции наиболее оптимальны для выполнения задачи. Диаграмма логической модели данных представлена на рисунке 16.

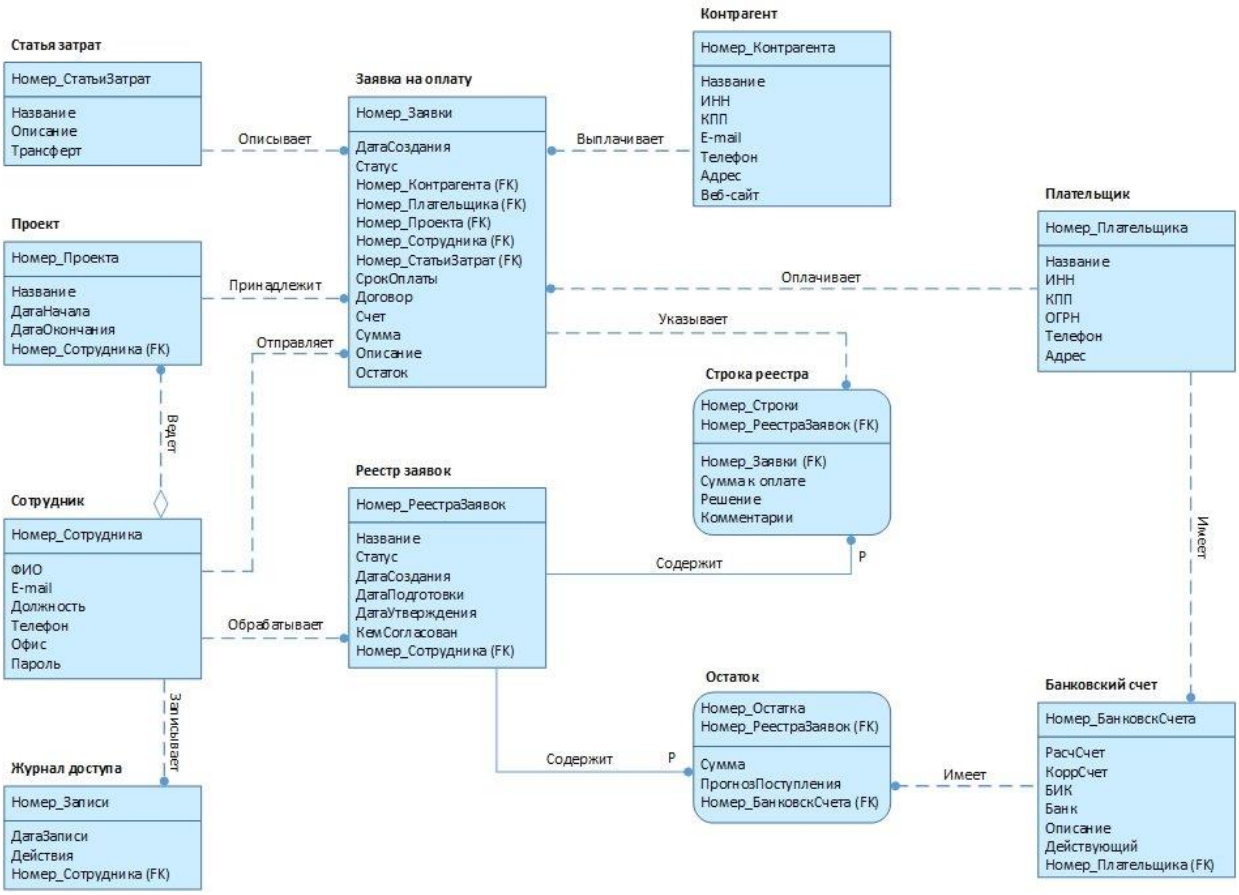


Рисунок 16 – Диаграмма логической модели данных

Все сущности на приведенной диаграмме, кроме «Строка реестра» и «Остаток», являются независимыми поскольку их первичные ключи не содержат внешних ключей других сущностей. Между сущностями «Заявка на оплату», «Статья затрат», «Контрагент», «Плательщик» и «Проект» установлена обязательная неидентифицирующая связь, поскольку в каждой заявке на оплату обязательно требуется указать эти значения. Связь между сущностями «Проект» и «Сотрудник» установлена необязательной и неидентифицирующей, поскольку не для каждого проекта может быть указан ответственный сотрудник. Зависимыми сущностями являются «Остаток» и «Строка реестра», поскольку их первичный ключ содержит внешний ключ «Реестра заявок». Кардинальность отношений между «Остаток», «Строка реестра» и сущностью «Реестр заявок» обозначена как P, поскольку каждый

реестр заявок должен содержать минимум одну или большее количество таких дочерних сущностей.

## 2.10 Требования к программно-аппаратному обеспечению

Согласно проектному решению система размещается на трех узлах: клиентский узел, сервер приложений и сервер базы данных. В таблице 9 представлены требования к программно-аппаратному обеспечению.

Таблица 9 – Требования к программно-аппаратному обеспечению

| Узел | Минимальные требования |
|---|---|
| Клиентская машина | Операционная система Windows 10 или Linux или macOS,<br>процессор Intel Core i3-2100,<br>ОЗУ 4 Гб,<br>HDD 60 Гб,<br>разрешение экрана не менее 1600x1024,<br>браузер Firefox или Chrome или Edge,<br>сетевая карта 10 Мбит. |
| Сервер приложений | Операционная система Windows Server 2012 или Linux,<br>процессор Intel Pentium 2.2 ГГц,<br>ОЗУ 8 Гб,<br>SSD: 80 Гб,<br>сетевая карта 100 Мбит,<br>фиксированный IPv4 адрес. |
| Сервер базы данных | Операционная система Windows Server 2012 или Linux,<br>процессор Intel Pentium 2.2 ГГц,<br>ОЗУ: 8 Гб,<br>SDD: 150 Гб,<br>сетевая карта 100 Мбит,<br>фиксированный IPv4 адрес. |

Клиентский узел — это рабочая станция на которой работает конечный пользователь АИС. Узел сервера приложений содержит программные компоненты веб-сервера и бизнес-логику системы. На узле базы данных размещается компоненты реляционной СУБД.

## 2.11 Прототипы UI/UX

При проектировании системы полезным средством визуализации конечного продукта выступают прототипы пользовательского интерфейса. Прототип пользовательского интерфейса представляет собой графический макет внешнего вида разрабатываемой информационной системы. Макет модуля «Заявки» представлен на рисунке 17.

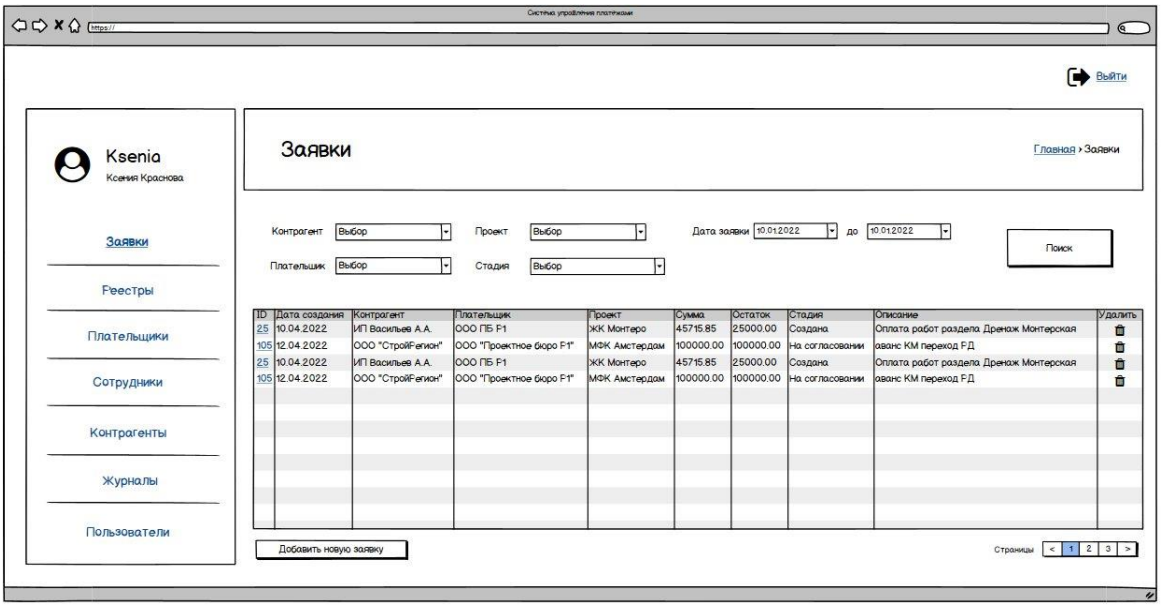


Рисунок 17 – Макет модуля «Заявки»

В левой части макета модуля «Заявки» находится навигационное меню, которое позволяет пользователю перейти в необходимый для работы модуль системы. В разделе «Заявки» отображается список заведенных в систему заявок на оплату с возможностью фильтрации по контрагенту, проекту, плательщику, стадии и дате создания. Если количество заявок превышает число строк, которые можно разместить на одном экране, то система будет разбивать вывод на страницы, номера которых расположены в правом нижнем углу. Макет формы работы с реестром показан на рисунке 18.

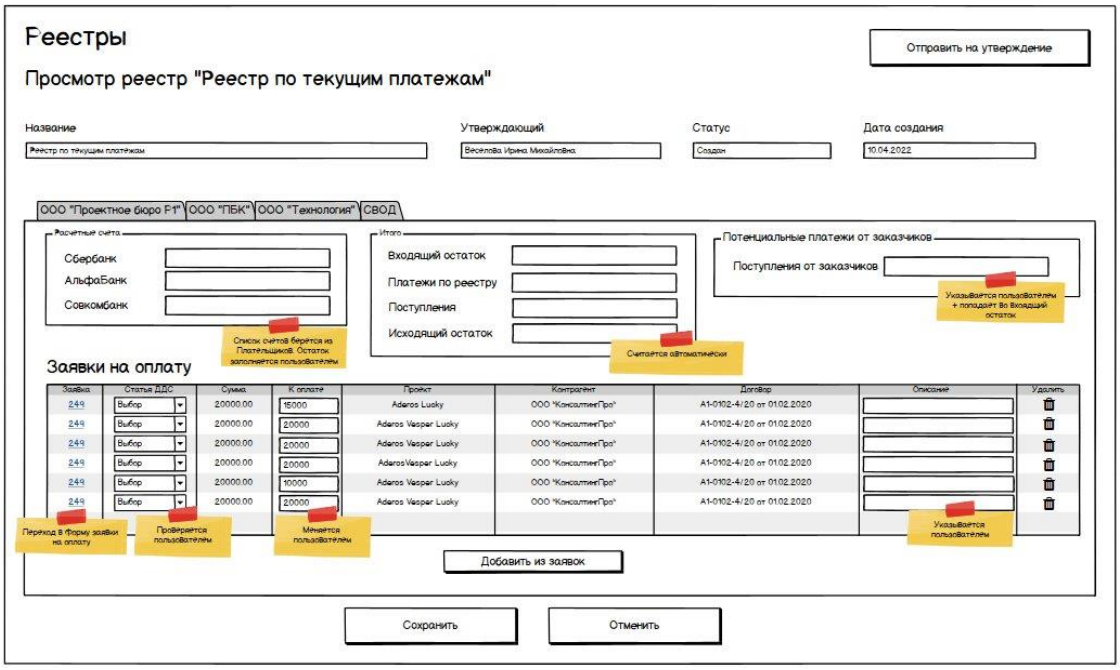


Рисунок 18 – Макет формы работы с реестром модуля «Реестры»

Каждая вкладка позволяет выбрать список заявок для оплаты по отдельному плательщику. По каждому плательщику пользователь указывает в панели «Расчетные счета» текущие остатки по банковским счетам выбранного плательщика. В поле «Поступления от заказчиков» пользователь указывает сумму, которая ожидается к поступлению от заказчиков. В списке заявок на оплату пользователь формирует перечень заявок, которые должны быть оплачены в рамках текущего реестра. По каждой заявке в реестре предусмотрена возможность изменить сумму к оплате и задать комментарий для утверждающего. При каждом изменении остатков по счетам, перечня или состава заявок на оплату система выполняет автоматический расчет и обновляет итоги в панели «Итоги». Для добавления новых заявок, пользователь нажимает кнопку «Добавить из заявок» и, в открывшейся форме, отмечает требуемые из представленного списка, ранее поданных на согласование заявок на оплату. После завершения формирования реестра, при нажатии на кнопку «Отправить на утверждение», реестр отправляется на согласование генеральному директору.

## Выводы по второй главе

Преимущества UML позволили улучшить качество проектных решений, сократить затрачиваемое время и повысить продуктивность. Применение диаграммы вариантов использования, дополненной текстовым описанием прецедентов для представления функционала системы, позволило продемонстрировать ключевые аспекты работы проектируемой системы. С помощью диаграммы классов идентифицированы основные сущности предметной области и их взаимосвязи. Диаграмма деятельности, выполненная для сквозного процесса оплаты по заявкам, продемонстрировала потоки управления и повысила понимание всего процесса. На диаграммах состояний удалось зафиксировать поведение и свойства, которыми обладают заявка на оплату и реестр заявок в зависимости от текущей стадии процесса оплаты. Логическая структура организации архитектуры АИС была показана на диаграмме компонентов, которая позволила отразить внешние интерфейсы, а также обозначить границы функциональности компонентов. Для размещения компонентов системы применяется схема с тремя узлами, которая обеспечивает надежность, отказоустойчивость и возможность дальнейшего масштабирования. При разработки концептуальной модели данных применялась графическая нотации П. Чена. Логическая модель данных выполнена по методологии IDEF1X. Учитывая приведенные требования к программно-аппаратному обеспечению система может быть успешно развернута в рабочей среде предприятия. Для лучшей визуализации системы применены прототипы пользовательского интерфейса.

# Глава 3 Разработка АИС «Payman»

## 3.1 Выбор технологии разработки

При выборе технологии разработки следует учитывать ряд критериев, включая следующие: масштаб проекта, инструментальная поддержка, обеспеченность готовыми библиотеками, наличие справочной документации, возможность интеграции и кроссплатформенность. Для оценки масштаба и сложности проекта можно использовать объем реализуемой программной логики и количество вовлеченных специалистов. Наличие библиотек и комфортная среда разработки с качественной инструментальной поддержкой позволяют существенно сократить временные затраты и повысить качество конечного решения. Кроме того, очень важно оценить перспективы развития и поддержки, поскольку применение стагнирующей технологии не позволит выполнять дальнейшую поддержку и обслуживание разработанной системы. Критерии сравнения и технологии представлены в таблице 10.

Таблица 10 – Сравнение технологий разработки

| Критерий | C# | Java | Python | PHP |
|---|---|---|---|---|
| Поддерживает ООП | 1 | 1 | 0,5 | 0,5 |
| Свободно распространяемый | 1 | 0,5 | 1 | 1 |
| Средства разработки | 1 | 0,5 | 0,5 | 0,5 |
| Кроссплатформенность | 1 | 1 | 1 | 1 |
| Платформа MVC | 1 | 1 | 0,5 | 1 |
| Интеграция с Active Directory | 1 | 1 | 0,5 | 0 |
| Интеграция с Power BI | 1 | 0,5 | 0 | 0 |
| Поддержка ORM | 1 | 1 | 1 | 1 |
| Поддержка ML | 0,5 | 0,5 | 1 | 0 |
| Итого | 8,5 | 7 | 6 | 5 |

По результатам сравнения, приведенным в таблице 10, для разработки ИС наиболее оптимальным выбором является C# и его экосистема, а для задач

машинного обучения Python и scikit-learn. Для реализации паттерна MVC будет использован свободно распространяемый кроссплатформенный ASP.NET Core MVC, который поддерживается и развивается компанией Microsoft.

### 3.2 Выбор СУБД

Выбор системы управления базой данных необходимо проводить с учетом выбранной ранее технологии разработки принимая во внимание критерии масштабируемости, кроссплатформенности, поддержки средств проектирования и надежности. Поскольку ранее в настоящей работе было принято проектное решение по использованию реляционной модели организации базы данных, то при сравнении будут оцениваться только наиболее популярные и поддерживающие данную модель решения. В таблице 11 представлено сравнение СУБД по критериям.

Таблица 11 – Сравнение СУБД

| Критерий | Microsoft SQL Server Express | PostgreSQL | MySQL |
|---|---|---|---|
| Кроссплатформенность | 1 | 1 | 1 |
| Бесплатно распространяемая | 1 | 1 | 1 |
| Интеграция Power BI | 1 | 0,5 | 0,5 |
| Поддержка и документация | 1 | 1 | 1 |
| Надежность | 1 | 1 | 1 |
| Масштабируемость | 0,5 | 1 | 0,5 |
| Интеграция с C# и .NET | 1 | 0,5 | 0,5 |
| Итого | 7,5 | 7 | 6 |

По результатам сравнения, приведенным в таблице 11, в разработке информационной системы будет использована Microsoft SQL Server Express. Применение этой СУБД позволяет получить бесплатную и качественную платформу для организации хранения данных обеспечивающую возможность

дальнейшего масштабирования и интеграции с обширной линейкой продуктов Microsoft. В качестве среды для проектирования и работы с SQL Server будет применяться Microsoft SQL Server Management Studio в редакции Express. Для взаимодействия с SQL Server удобно использовать расширение языка запросов Transact-SQL, который допускает применение переменных, управляющих конструкций и операторов, обработку и выполнение хранимых процедур и системных функций непосредственно на сервере базы данных.

### 3.3 Физическая модель данных

Физическая модель данных является модифицированной версией логической модели данных при построении которой учитываются особенности выбранной реализации СУБД. На рисунке 19 представлена диаграмма физической модели данных в нотации IDEF1X.

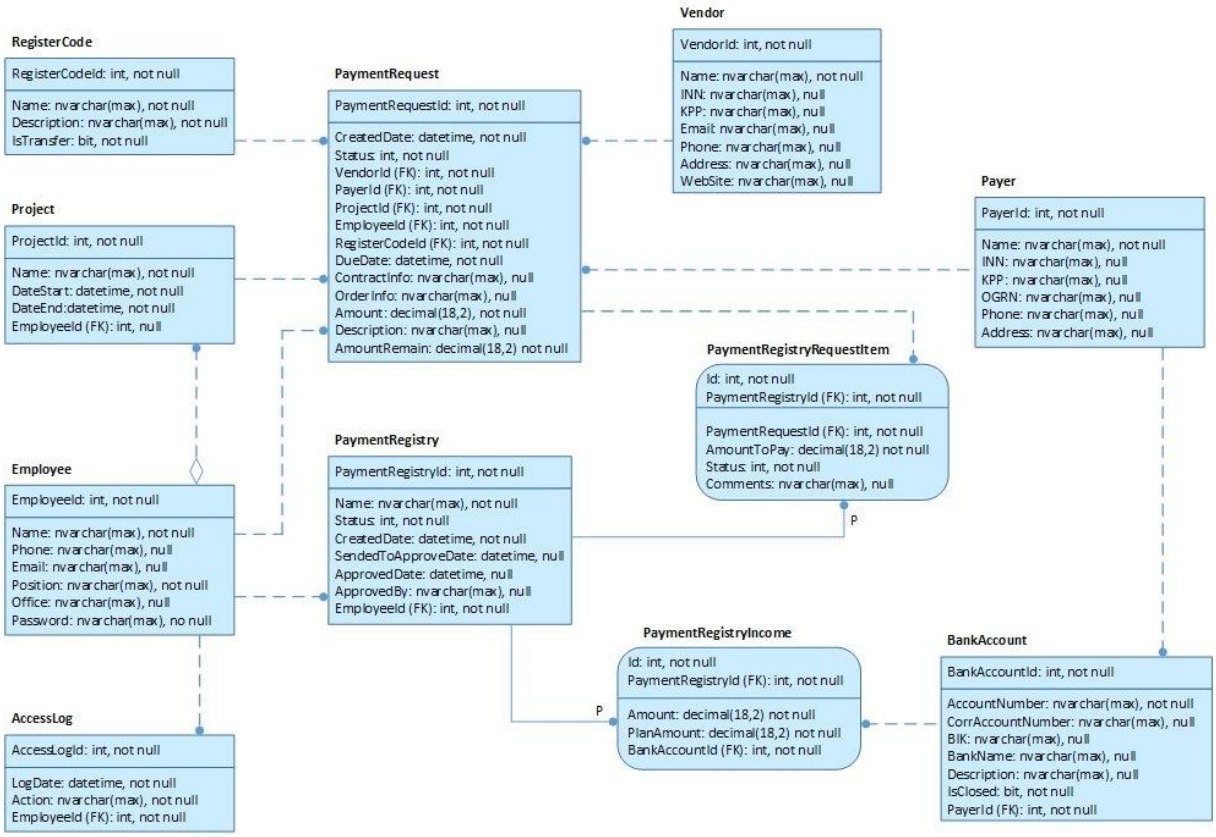


Рисунок 19 – Диаграмма физической модели данных

Как видно из диаграммы, представленной на рисунке 19, физическая модель данных содержит информацию о типах и включает имена сущностей и названия их атрибутов на английском языке, согласно стандарту именования, который применяется в выбранной СУБД. Можно заметить, что построение физической модели данных на основании логической модели данных не привело к изменениям в имеющихся взаимосвязях, составе сущностей и их отношениях. Для уточнения соответствия названия атрибутов сущностей следует сопоставить данную диаграмму с диаграммой логической модели данных.

### 3.4 Хранение данных и ORM

Для работы с данными необходимо сформировать структуру и задать типы данных с которыми будет работать система. В рамках данной работы будет использован подход CodeFirst. Этот подход подразумевает создание POCO классов для сущностей, которые являются отображением таблиц и содержат данные получаемые из СУБД. Связывание и обработку данных будет выполнять ORM Entity Framework Core. Для ее работы потребуется определить пользовательский класс, наследуемый от класса DbContext. В этом классе должны быть описаны все сущности, которые будут обрабатываться ORM. Основными POCO классами в рамках разрабатываемой системы будут:

- класс AccessLog представляет сущность «Журнал доступа»;
- класс BankAccount представляет сущность «Банковский счет»;
- класс Employee представляет сущность «Сотрудник»;
- класс Payer представляет сущность «Плательщик»;
- класс PaymentRegistry представляет сущность «Реестр заявок»;
- класс PaymentRequest представляет сущность «Заявка на оплату»;
- класс Project представляет сущность «Проект»;
- класс RegisterCode представляет сущность «Статья затрат»;
- класс Vendor представляет сущность «Контрагент».

Для первоначального наполнения базы данных потребуется реализовать специальный метод SeedDefault. Этот метод будет проверять существование базы данных и готовность миграций схемы данных. POCO классы сущностей необходимо указать в качестве параметризированного типа DbSet в свойствах контекстного класса, который наследуется от DbContext. Код контекстного класса EFDatabaseContext представлен на рисунке 20.

```
public class EFDatabaseContext : IdentityDbContext<PaymanUser>
{
    public EFDatabaseContext(DbContextOptions<EFDatabaseContext> options) : base(options) { }
    protected override void OnModelCreating(ModelBuilder builder)...
    public DbSet<RegisterCode> RegsiterCodes { get; set; }
    public DbSet<Vendor> Vendors { get; set; }
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Payer> Payers { get; set; }
    public DbSet<BankAccount> BankAccounts { get; set; }
    public DbSet<Project> Projects { get; set; }
    public DbSet<PaymentRequest> PaymentRequests { get; set; }
    public DbSet<PaymentRegistry> PaymentRegistries { get; set; }
    public DbSet<PaymentRegistryIncome> PaymentRegistryIncomes { get; set; }
    public DbSet<PaymentRegistryItemRequest> PaymentRegistryRequestItems { get; set; }
    public DbSet<AccessLog> AccessLogs { get; set; }
}
```

Рисунок 20 – Фрагмент кода контекстного класса EFDatabaseContext

По своей сути, POCO класс является обычным классом с определенными полями и свойствами, который может обладать поведением. Поля и свойства POCO классов обычно соответствуют модели данных. Код одного из таких классов представлен на рисунке 21.

```
public class BankAccount
{
    public int BankAccountId { get; set; }
    public string AccountNumber { get; set; }
    public string CorrAccountNumber { get; set; }
    public string BIK { get; set; }
    public string BankName { get; set; }
    public string Description { get; set; }
    public bool IsClosed { get; set; }
    public int PayerId { get; set; }
}
```

Рисунок 21 – Фрагмент кода POCO класса BankAccount

При запуске системы, для инициализации контекстного класса данных, потребуется выполнить конфигурацию Entity Framework Core. В специальном классе Startup, который отвечает за общую настройку системы, необходимо настроить класс EFDatabaseContext. Для этого потребуется вызвать метод AddDbContext в методе ConfigureServices и указать параметры подключения к серверу баз данных, а также другие опции для настройки работы контекстного класса. Код конфигурации контекстного класса показан на рисунке 22.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddDbContext<EFDatabaseContext>(opts => {
        opts.UseSqlServer(Configuration["ConnectionStrings:DatabaseConnection"]);
        opts.EnableSensitiveDataLogging(Configuration["DBSensitiveLoggingEnabled"] == "True");
    });
    services.AddDistributedMemoryCache();
    services.AddSession();
    services.AddIdentity<PaymanUser, IdentityRole>(options =>
    {
        options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromDays(5);
    }).AddEntityFrameworkStores<EFDatabaseContext>();
    services.AddTransient<ISeedData, EFSeedData>();
}
```

Рисунок 22 – Фрагмент кода метода ConfigureServices

После того, как создан контекстный класс и произведена настройка подключения следуют выполнить миграцию. В процессе миграции исследуется структура и взаимосвязи контекста, а также строятся требуемые зависимости для работы ORM. Для запуска миграции в консоли Package Manager необходимо выполнить команду Add-Migration с указанием имени миграции. Миграция создает специальные классы, которые размещаются в директории Migrations. Каждая миграция создает отдельный файл, который содержит временную метку и название миграции. После того как файлы миграции созданы необходимо выполнить команду для синхронизации миграции с сервером базы данных. Для этого, в консоли Package Manager, нужно выполнить команду Update-Database. Когда команда будет успешно завершена, на сервере базы данных будет создана структура таблиц и полей, которая отражает требуемую схему классов.

### 3.5 Оформление и шаблоны

Для работы системы требуется создать оформление и работоспособную верстку с применением HTML разметки и каскадных таблиц стилей для каждой экранной формы с которой взаимодействует пользователь в процессе эксплуатации. В качестве базового набора стилей и форматирования будет использоваться шаблон Inspinia. В рамках разрабатываемой системы для рендеринга и отображения страниц применен движок Razor, который поставляется совместно с ASP.NET Core. Перечень шаблонов представлений показан на рисунке 23.

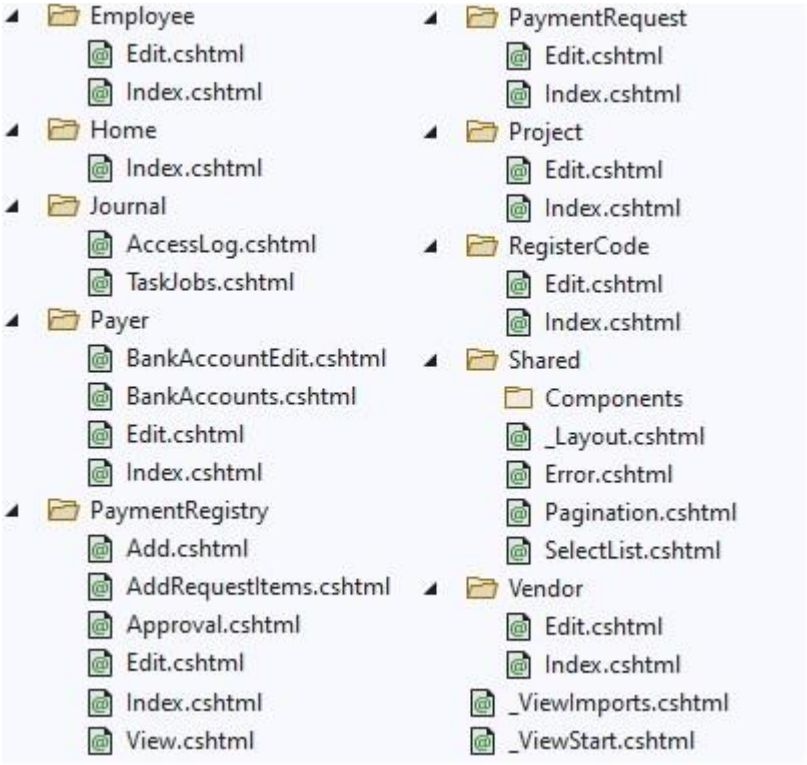


Рисунок 23 – Шаблоны представлений Razor

Все шаблоны представлений располагаются в директории Views. Каждому контроллеру сопоставляется дочерняя директория внутри директории Views с именем, который соответствует имени контроллера за исключением опущенного окончания Controller. Так, например, для

контроллера ProjectController сопоставляется директория Project. Внутри директории Project содержаться непосредственные шаблоны представления Razor, которые вызываются при работе контроллера для вывода пользователю. Также в директории Views содержаться два специализированных шаблона:

- шаблон _ViewImports.cshtml содержит инструкции по импорту пространств имен во все шаблоны представления;
- шаблон _ViewStart.cshtml размещает программный код, выполняемый перед обработкой кода других шаблонов представления.

Для унификации шаблонов представлений используется специальный шаблон _Layout.cshtml, который располагается в директории Shared. Этот шаблон представления содержит код, который должен быть повторен в каждом шаблоне представлении.

### 3.6 Контроль доступа

Согласно проектному решению, необходимо гарантировать ограниченный доступ как в саму систему, так и к отдельным компонентам. Указанную задачу можно разбить на две составляющие. Первая составляющая подразумевает реализацию авторизации пользователя для непосредственного входа в систему. Вторая составляющая включает управление пользователями внутри системы. Для целей контроля доступа необходимо создать контроллер AccountController, который будет отвечать за все программную логику взаимодействия с пользователем. Экземпляр контекста базы данных класс AccountController получает в качестве параметра в конструкторе класса с помощью механизма Dependency Injection, который реализован ASP.NET Core MVC.

Метод Login получает введенные пользователем данные и выполняет их сверку с теми, что хранятся в базе данных. Если пользователь ввел корректный логин и пароль, то в данные сессии пользователя заносится флаг успешной

авторизации и записывается факт входа в журнал доступа. Если пользователь указал неподходящие данные, то выводится повторно форма для ввода логина и пароля. Код метода Login в классе AccountController показан на рисунке 24.

```
[AllowAnonymous]
public ViewResult Login(string returnUrl)
{
    return View(new LoginViewModel() { ReturnUrl = returnUrl });
}

[HttpPost]
[ValidateAntiForgeryToken]
[AllowAnonymous]
public async Task<IActionResult> Login(LoginViewModel viewModel)
{
    if (ModelState.IsValid)
    {
        var log = new AccessLog();
        log.LogDate = DateTime.Now;
        string login = viewModel.Name;
        string remoteIPAddress = Request.HttpContext.Connection.RemoteIpAddress.ToString();
        var user = await _userManager.FindByNameAsync(viewModel.Name);
        if (user != null)
        {
            await _signInManager.SignOutAsync();
            var res = await _signInManager.PasswordSignInAsync(user, viewModel.Password, false, true);
            if (res.Succeeded)
            {
                log.Action = $"Авторизация {login} с IP {remoteIPAddress} выполнена успешно";
                _dbContext.AccessLogs.Add(log);
                _dbContext.SaveChanges();
                return Redirect(viewModel.ReturnUrl ?? "/Home");
            }
        }
        ModelState.AddModelError("Auth", "Невозможно завершить операцию");
        log.Action = $"Авторизация {login} с IP {remoteIPAddress} провалена";
        _dbContext.AccessLogs.Add(log);
        _dbContext.SaveChanges();
    }
    return View(viewModel);
}
```

Рисунок 24 – Фрагмент кода авторизации в AccountController

Кроме авторизации класс AccountController выполняет задачи по управлению пользователями. Для этой цели реализованы следующие методы: метод Index отображает список пользователей, метод Logout позволяет пользователю завершить текущую сессию, метод Edit позволяет изменять пользователей, метод Delete позволяет удалять пользователей из системы, метод ChangePassword позволяет изменять пароль.

Для активации функционала сессий необходимо добавить в метод ConfigureServices класса Startup вызов метода AddDistributedMemoryCache для

IServiceCollection, а также зарегистрировать провайдер обработки данных по умолчанию с помощью метода AddSession. В методе конфигурирования приложения Configure класса Startup необходимо вызвать метод UseSession для IApplicationBuilder, который активирует обработчик данных сессий пользователей. Для обеспечения контроля доступа в модулях системы следует применить атрибут Authorize.

### 3.7 Модули системы

Для работы системы необходимо реализовать функционал по работе со справочниками. В качестве примера будет рассмотрен модуль контрагенты. Необходимо обеспечить возможность добавления, просмотра и удаления контрагентов. Класс VendorController содержит логику взаимодействия с пользователем при работе с контрагентами. Он помечен атрибутом Authorize и атрибутом UILeftMenu, который обеспечивает отображение в левом навигационном меню системы. Конструктор класса получает экземпляр контекстного класса EFDatabaseContext для взаимодействия с базой данных посредством ORM и экземпляр класса UserManager для взаимодействия с подсистемой Identity. Код класса показан на рисунке 25.

```
[Authorize]
[UILeftMenu(nameof(VendorController), "Контрагенты", "fa-truck", 3)]
public class VendorController : UIController
{
    public override string UICaption { get; protected set; } = "Контрагенты";
    private readonly EFDatabaseContext _dbContext;

    public VendorController(EFDatabaseContext dbContext, UserManager<PaymanUser> userManager)...

    public ViewResult Index(string filterName, string filterINN, int currentPage = 0)...

    public IActionResult Edit(int id)...

    [HttpPost]
    public IActionResult Edit(Vendor vendor)...

    protected bool CheckValidVendorFields(string name, string INN, int id = 0)...

    [HttpPost]
    public RedirectToActionResult Delete(int id)...
}
```

Рисунок 25 – Фрагмент кода класса VendorController

При выводе используются шаблоны, которые находятся в директории Views/Vendor. Страница списка контрагентов показана на рисунке 26.

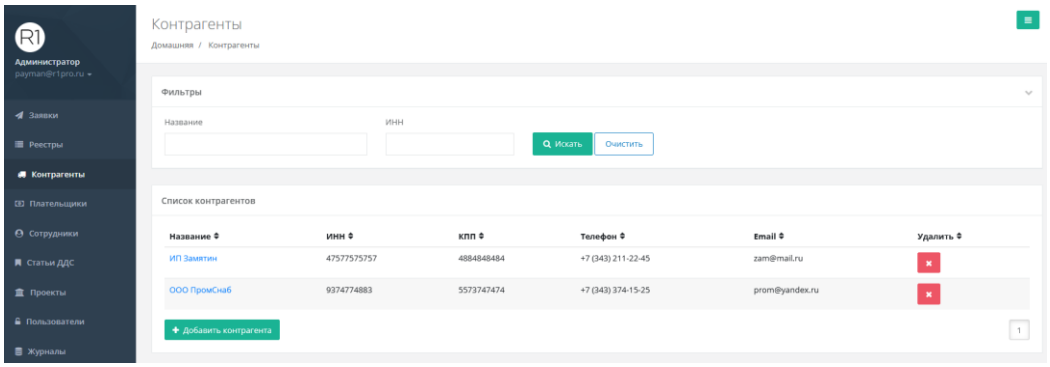


Рисунок 26 – Страница списка контрагентов

Класс VendorController реализует следующие методы: метод Index для вывода списка контрагентов, метод Edit для вывода и обработки данных из формы изменения, метод CheckValidVendorFields для проверки правильности и отсутствия дубликатов, метод Delete для удаления контрагента. Страница для работы с контрагентом представлена на рисунке 27.

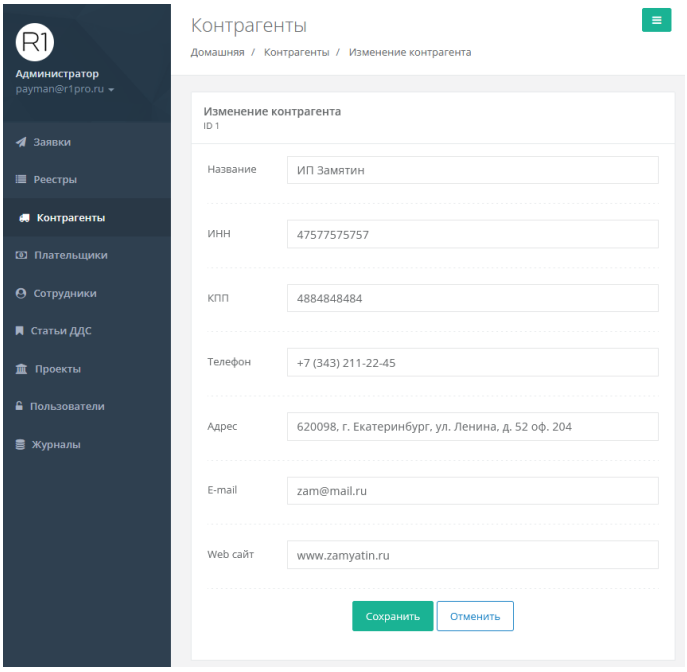


Рисунок 27 – Форма для работы с карточкой контрагента

Другие модули системы, которые обеспечивают работу справочников, имеют схожую реализацию, формы вывода и структуру программного кода.

Для работы пользователей с заявками на оплату разработан класс PaymentRequestController. Код класса PaymentRequestContoller показан на рисунке 28.

```
[Authorize]
[UILeftMenu(nameof(PaymentRequestController), "Заявки", "fa-send", 1)]
public class PaymentRequestController : UIController
{
    public override string UICaption { get; protected set; } = "Заявки";
    private readonly EFDatabaseContext _dbContext;

    public PaymentRequestController(EFDatabaseContext dbContext, UserManager<PaymanUser> userManager) ...

    public ViewResult Index(PaymentRequestIndexViewModel.SearchFilters filters, int currentPage = 0)...

    public IActionResult Edit(int id)...

    [HttpPost]
    public IActionResult Edit(PaymentRequest request)...

    protected bool CheckValidRequestFields(int vendorId, int payerId, int employeeId, DateTime dueDate, int id = 0)...

    [HttpPost]
    public RedirectToActionResult Delete(int id)...
}
```

Рисунок 28 – Фрагмент кода класса PaymentRequestController

В качестве параметров конструктор класса получает экземпляр класса EFDatabaseContext и экземпляр класса UserManager. Для обеспечения контроля доступа класс помечен атрибутом Authorize, а также атрибутом UILeftMenu для отображения класса в левом навигационном меню. Класс реализует следующие методы: метод Index для вывода списка заявок, метод Edit для вывода формы и обработки переданных данных, метод CheckValidRequestFields для проверки корректности и метод Delete для удаления заявки.

При работе со списком заявок обеспечивается возможность поиска, фильтрации и, при большом количестве записей, вывод по страницам. Для передачи в метод Index параметров поиска и фильтрации, которые заданы пользователем, используется класс PaymentRequestIndexViewModel. Код класса показан на рисунке 29.

```
public class PaymentRequestIndexViewModel
{
    public SearchFilters Filters { get; set; } = new SearchFilters();
    public List<PaymentRequestItemViewModel> Requests { get; set; } = new List<PaymentRequestItemViewModel>();
    public PaginationViewModel Pagination { get; set; } = new PaginationViewModel();

    public class SearchFilters
    {
        public DateTime? CreatedDateStart { get; set; }
        public DateTime? CreatedDateEnd { get; set; }
        public int? VendorId { get; set; }
        public int? PayerId { get; set; }
        public int? ProjectId { get; set; }
        public PaymentRequestStatus? Status { get; set; }
    }
}
```

Рисунок 29 – Фрагмент кода класса PaymentRequestIndexViewModel

Данный класс содержит вложенный класс SearchFilters, который определяет поля для поиска и фильтрации списка. Для постраничного вывода применяется класс PaginationViewModel. Страница со списком заявок на оплату представлена на рисунке 30.

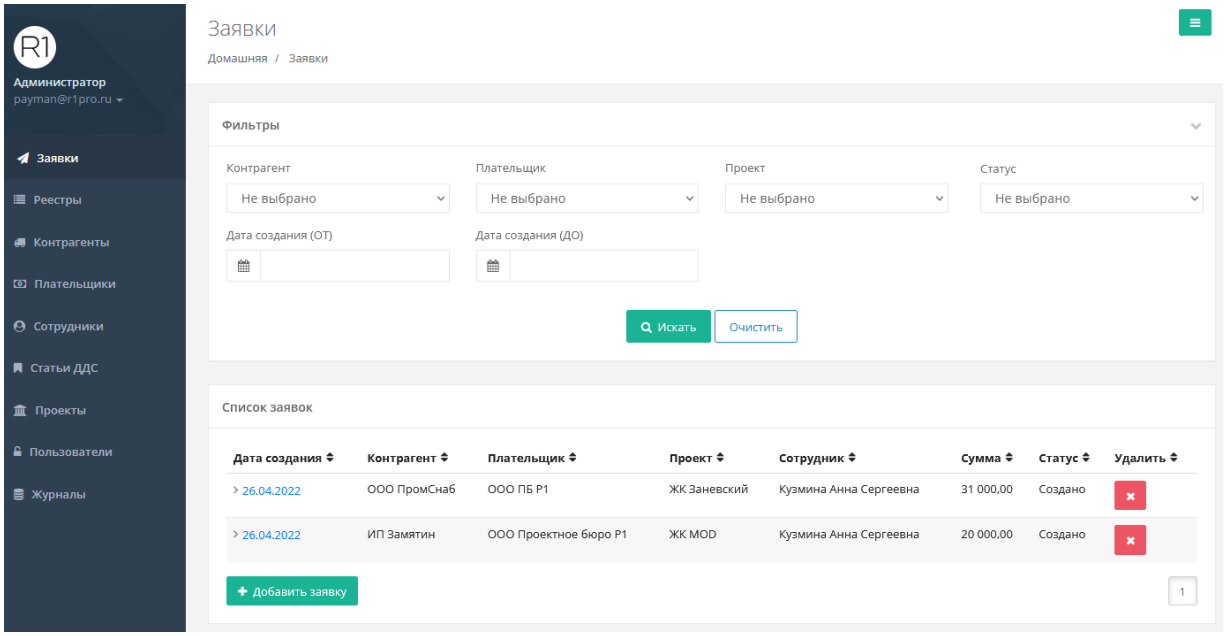


Рисунок 30 – Страница со списком заявок на оплату

Для работы пользователей с реестрами заявок на оплату реализован класс PaymentRegistryController. Код класса показан на рисунке 31.

```
[Authorize]
[UILeftMenu(nameof(PaymentRegistryController), "Реестры", "fa-list", 2)]
public class PaymentRegistryController : UIController
{
    public override string UICaption { get; protected set; } = "Реестры";
    private readonly EFDatabaseContext _dbContext;
    public PaymentRegistryController(EFDatabaseContext dbContext, UserManager<PaymanUser> userManager)...
    public ViewResult Index(PaymentRegistryIndexViewModel.SearchFilters filters, int currentPage = 0)...
    public ViewResult Add()...
    [HttpPost]
    public IActionResult Add(PaymentRegistryAddViewModel viewModel)...
    public IActionResult Edit(int id, int payerId = 0)...
    [HttpPost]
    public IActionResult Edit(int id, int payerId, Dictionary<int, decimal> bankAccounts,
                              Dictionary<int, decimal> customerIncomes,
                              Dictionary<int, decimal> amountsToPay,
                              Dictionary<int, int?> registerCodes)...
    [HttpPost]
    public string EditCustomerIncomeAmount(int payerId, decimal amount)...
    [HttpPost]
    public string EditBankAccountAmount(int bankAccountId, decimal amount)...
    [HttpPost]
    public string EditRequestComments(int requestId, string comments)...
    [HttpPost]
    public string EditRequestAmountToPay(int requestId, decimal amount)...
    [HttpPost]
    public string EditRequestStatus(int requestId, PaymentRegistryItemStatus status)...
    [HttpPost]
    public string EditPaymentRequestRegisterCode(int requestId, int? registerCodeId)...
    protected PaymentRegistryViewModel FillViewModel(PaymentRegistry registry)...
    protected string GetTargetActionByStatus(PaymentRegistryStatus status)...
    public IActionResult AddRequestItems(int id, int payerId)...
    [HttpPost]
    public IActionResult AddRequestItems(int id, int payerId, List<int> selectedRequests)...
    public RedirectToActionResult DeleteRequestItem(int id, int registryId, int payerId)...
    protected void CalculateTransfers(int registryId)...
    public IActionResult SendToApprove(int id)...
    [HttpPost]
    public RedirectToActionResult Delete(int id)...
    public IActionResult Approval(int id, int payerId = 0)...
    [HttpPost]
    public IActionResult Approval(int id, int payerId, Dictionary<int, decimal> requestAmountsToPay,
                                  Dictionary<int, string> requestComments,
                                  Dictionary<int, PaymentRegistryItemStatus> requestStatus)...
    public IActionResult Approve(int id)...
    public IActionResult SendBackToEdit(int id)...
    public IActionResult View(int id)...
}
```

Рисунок 31 – Фрагмент кода класса PaymentRegistryController

Класс помечен атрибутом Authorize для обеспечения контроля доступа пользователей, а также атрибутом UILeftMenu для отображения в левом навигационном меню. Через параметры конструктора передаются экземпляры классов EFDatabaseContext и UserManager для работы с базой данных с помощью ORM и взаимодействия с подсистемой Identity. Класс реализует методы, указанные в таблице 12.

Таблица 12 – Описание методов класса PaymentRegistryContoller

| Метод | Описание |
|---|---|
| Index | Выводит форму со списком реестров заявок |
| Add | Выводит форму для добавления нового реестра и обрабатывает переданные данные из формы для добавления |
| Edit | Выводит форму и обрабатывает изменения в реестре |
| EditCustomerIncomeAmount | Поддерживает изменение ожидаемого поступления денежных средств посредством Ajax |
| EditPaymentRequestRegisterCode | Поддерживает изменение статьи затрат для заявки на оплату посредством Ajax |
| EditBankAccountAmount | Поддерживает изменение остатка по банковскому счету посредством Ajax |
| EditRequestComments | Поддерживает изменение комментария по заявке на оплату посредством Ajax |
| EditRequestAmountToPay | Поддерживает изменение суммы к оплате по заявке посредством Ajax |
| EditRequestStatus | Поддерживает изменение решения по заявке на оплату посредством Ajax |
| FillViewModel | Формирует модель представления |
| GetTargetActionByStatus | Определяет возможные действия с реестром исходя из его текущего статуса |
| AddRequestItems | Выводит форму и обрабатывает данные для добавления заявок на оплату в реестр |
| DeleteRequestItem | Удаляет выбранную заявку из реестра |
| CalculateTransfers | Выполняет расчеты по перемещениям денежных средств между плательщиками внутри реестра |
| SendToApprove | Отправляет реестр заявок на согласование |
| Delete | Выполняет удаление реестра заявок |
| Approval | Выводит форму для согласования реестра заявок и обрабатывает переданные данные с внесенными изменениями |
| Approve | Выполняет утверждение реестра заявок |
| SendBackToEdit | Выполняет возврат реестра заявок на доработку |
| View | Выводит форму отображения реестра заявок, которая не позволяет вносить изменения |

Для вывода класс PaymentRegistryController использует следующие шаблоны: Add.cshtml для формы добавления реестра, AddRequestItems.cshtml для формы добавления заявок на оплату, Approval.cshtml для формы согласования реестра, Edit.cshml для формы редактирования реестра, Index.cshtml для вывода списка реестров и View.cshtml для отображения без

возможности внесения изменений. Страница с формой для работы с реестром заявок показана на рисунке 32.

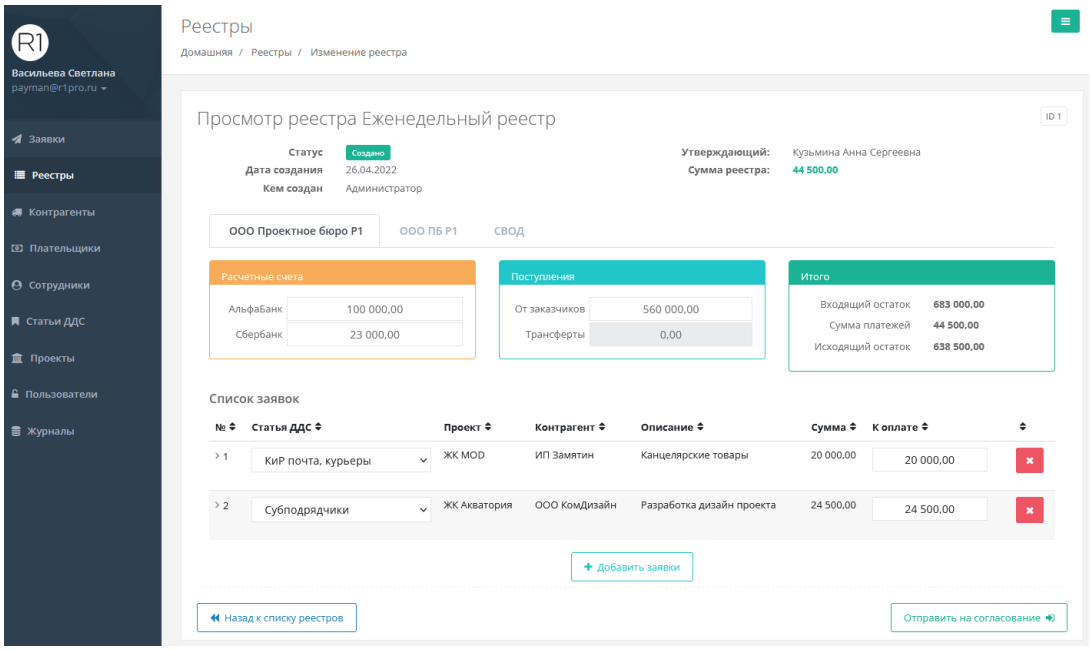


Рисунок 32 – Страница с формой для работы с реестром заявок

После любых внесенных изменений в реестр, система выполняет автоматический расчет сводных показателей по всем плательщикам. Страница с итоговой информацией по реестру показана на рисунке 33.

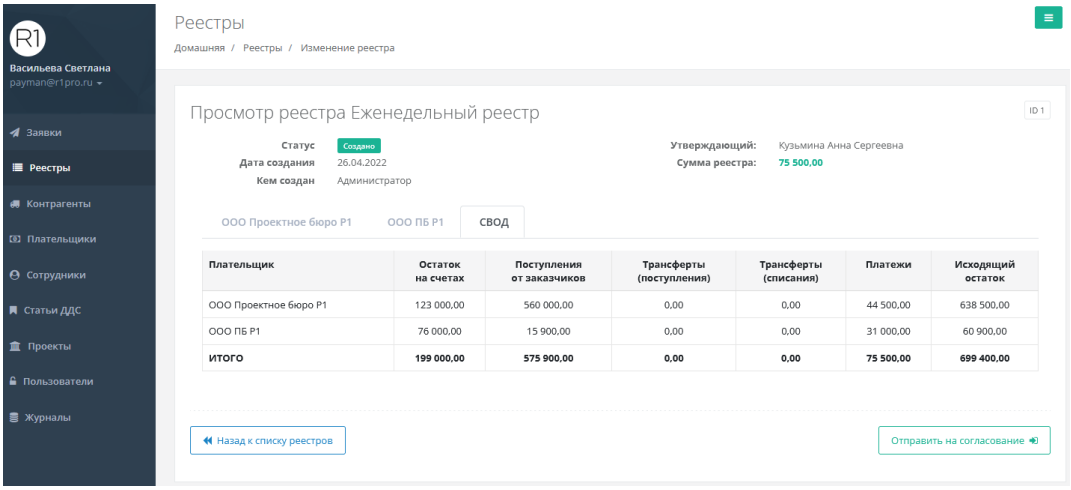


Рисунок 33 – Страница со сводной информацией по реестру заявок

Формирование согласованного реестра платежей обеспечивает класс RegistryExcelPrinter используя библиотеку EPPlus. Код класса показан на рисунке 34.

```
public class RegistryExcelPrinter
{
    protected void PrintHeader(ExcelWorksheet sheet, string companyName, DateTime date)...
    protected int PrintBankAccounts(ExcelWorksheet sheet, List<IncomeRow> accounts, decimal payments)...
    public void PrintPayments(ExcelWorksheet sheet, List<PaymentRow> payments, int startRow)...
    protected void PrintSummary(ExcelWorksheet sheet, List<SummaryRow> summary)...
    protected void MakeTableHeaderStyle(ExcelRange range)...
    protected void MakeBorders(ExcelRange range)...
    public void Print(string fileName, Workbook book)...
    public class IncomeRow...
    public class PaymentRow...
    public class SummaryRow...
    public class Sheet...
    public class Workbook...
}
```

Рисунок 34 – Фрагмент кода класса RegistryExcelPrinter

Для запуска выгрузки согласованного реестра необходимо передать в метод Print имя файла, в который следует выгрузить реестр, и данные для которых выполняется формирование.

### 3.8 Алгоритмы машинного обучения

Согласно учетной политики, заявки должны быть классифицированы соответствующей статьей затрат. Для обучения моделей машинного обучения необходимо выполнить сбор, систематизацию и анализ исходных данных. Исходные данные должны обеспечивать репрезентативность и полноту выборки [18]. Для целей настоящей работы было собрано 4395 образцов ранее поданных заявок на оплату за прошедшие периоды. Каждый образец был обработан и классифицирован. Основными признаками для обучения моделей выбраны следующие: дата создания заявки, название контрагента, название проекта, ФИО сотрудника, описание заявки, сумма и статья затрат.

Задача классификации может быть решена с применением различных алгоритмов машинного обучения. Для экспериментов были выбраны:

– деревья принятия решений – иерархическая модель, состоящая из узлов, где каждый узел содержит подмножество исходной выборки;
– k-ближайших соседей – метрический алгоритм, построенный на идеи поиска среди уже известных образцов в исходной выборке;
– логистическая регрессия – алгоритм, основанный на статистической модели регрессии с помощью введения логит-функции;
– наивный байесовский классификатор – алгоритм для работы с текстовыми данными, который основан на теореме Байеса.

Выбор алгоритмов обусловлен различиями в принципах организации и хорошими показателями предсказаний. Для преобразования числовых данных использована min-max нормализация в диапазон [0;1]. Преобразование категориальных признаков выполнялось с помощью one-hot encoding. Для кодирования слов в векторном виде применялся метод мешка слов. Для преобразования слов в текстовом описании использовался морфологический анализатор Pymorphy2 [20]. В таблице 13 приведены результаты предсказаний, которые получены при обучении на следующих признаках: название контрагента, название проекта, ФИО сотрудника и сумма к оплате.

Таблица 13 – Результаты предсказаний без текстового описания

| Алгоритм | Accuracy | Precision | Recall |
|---|---|---|---|
| Деревья принятия решений | 0,95 | 0,95 | 0,95 |
| K-ближайших соседей | 0,93 | 0,93 | 0,93 |
| Логистическая регрессия | 0,94 | 0,95 | 0,94 |

Алгоритмы продемонстрировали хорошие показатели предсказаний. Однако, если новые заявки будут содержать признаки, которые ранее не встречались в выборке, то это может привести к ухудшению качества прогноза. Соответственно, следует учесть и текстовое описание заявки. Результаты обучения только по текстовым данным с использованием наивного байесовского классификатора приведены в таблице 14.

Таблица 14 – Результаты наивного байесовского классификатора

| Модель | Accuracy | Precision | Recall |
|---|---|---|---|
| Бернулли | 0,70 | 0,70 | 0,70 |
| Мультиномиальная модель | 0,88 | 0,88 | 0,88 |

Получив оценки предсказаний можно выбрать наиболее релевантные модели и сформировать суррогатную выборку по принципу, который показан на рисунке 35.

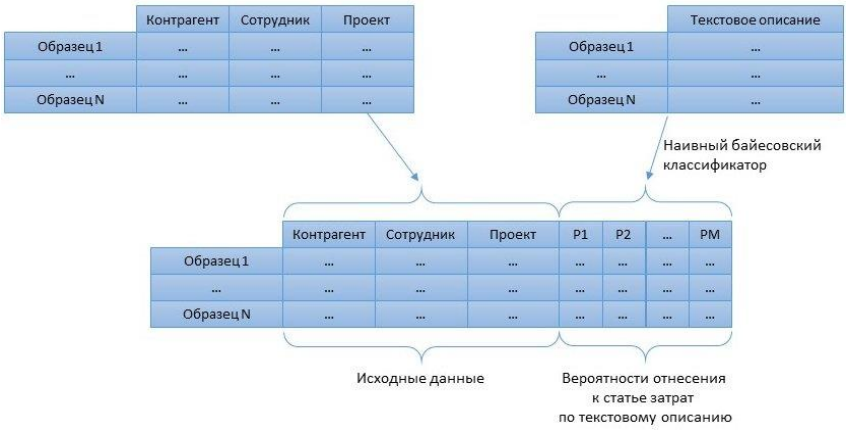


Рисунок 35 – Схема формирования суррогатной выборки

Результаты предсказаний алгоритмов машинного обучения, которые были получены по суррогатной выборке, представлены в таблице 15.

Таблица 15 – Результаты предсказаний по суррогатным данным

| Алгоритм | Accuracy | Precision | Recall |
|---|---|---|---|
| Деревья принятия решений | 0,95 | 0,95 | 0,95 |
| K-ближайших соседей | 0,97 | 0,97 | 0,97 |
| Логистическая регрессия | 0,96 | 0,96 | 0,96 |

Результаты показали, что примененный способ создания суррогатной выборки позволил улучшить качество предсказаний. Итогами проделанной

работы стали созданные и обученные модели машинного обучения, которые можно применить для классификации заявок на оплату. Программный код классификации заявок приведен в Приложении В.

### 3.9 Оценка экономической эффективности

Для проведения анализа экономической эффективности были собраны сводные данные за последние 12 месяцев работы предприятия. До внедрения ИС была подана и обработана 2581 заявка на оплату, а также сформировано и утверждено 96 реестров. В таблице 16 представлен расчет временных затрат.

Таблица 16 – Временные затраты до внедрения ИС в месяц

| Роль | Всего заявок | | Всего реестров | Среднее время | Суммарное время |
|---|---|---|---|---|---|
| | Подано | Обработано | | | |
| Инициатор заявок | 215 | - | - | 5 мин. | 18 ч. |
| Сотрудник экономической службы | - | 215 | - | 10 мин. | 36 ч. |
| Начальник экономической службы | - | - | 8 | 60 мин. | 8 ч. |

Оценка временных затрат в денежном эквиваленте показана в таблице 17.

Таблица 17 – Стоимостные затраты до внедрения ИС в месяц

| Роль | Средняя зарплата | Стоимость норма-часа | Затраченное время | Сумма затрат |
|---|---|---|---|---|
| Инициатор заявок | 50 000 руб. | 313 руб. | 18 часов | 5 634 руб. |
| Сотрудник экономической службы | 50 000 руб. | 313 руб. | 36 часов | 11 268 руб. |
| Начальник экономической службы | 80 000 руб. | 500 руб. | 8 часов | 4 000 руб. |
| | | | Итого в месяц | 20 902 руб. |

Таким образом, как показывают данные в таблице 17, на обеспечение бизнес-процесса управления платежами предприятие затрачивает в денежном

эквиваленте более 20 тысяч рублей ежемесячно. В результате ввода ИС в опытную эксплуатацию были получены оценки затраченного рабочего времени, которые представлены в таблице 18.

Таблица 18 – Сравнение показателей затрачиваемого времени

| Роль | Среднее время на единицу | | Суммарное время | |
|---|---|---|---|---|
| | Без ИС | ИС | Без ИС | ИС |
| Инициатор заявок | 5 мин. | 3 мин. | 18 ч. | 11 ч. |
| Сотрудник экономической службы | 10 мин. | 0 мин. | 36 ч. | 0 ч. |
| Начальник экономической службы | 60 мин. | 30 мин. | 8 ч. | 4 ч. |

На основании новых оценок временных затрат сотрудников, в рамках бизнес-процесса, при опытной эксплуатации информационной системы можно выполнить расчет сокращения затрат. Расчет стоимостных затрат при внедрении ИС приведен в таблице 19.

Таблица 19 – Стоимостные затраты при внедрении ИС в месяц

| Роль | Средняя зарплата | Стоимость норма-часа | Затраченное время | Сумма затрат |
|---|---|---|---|---|
| Инициатор заявок | 50 000 руб. | 313 руб. | 11 часов | 3 443 руб. |
| Сотрудник экономической службы | 50 000 руб. | 313 руб. | 0 часов | 0 руб. |
| Начальник экономической службы | 80 000 руб. | 500 руб. | 4 часа | 2 000 руб. |
| | | | Итого в месяц | 5 443 руб. |

Представленные в таблице 19 данные показывают, что внедрение ИС позволит предприятию сэкономить на трудовых затратах 15 тысяч 459 рублей, то есть более 70% ежемесячно. Оценка затрат на разработку и внедрение ИС приведена в таблице 20.

Таблица 20 – Оценка затрат на разработку и внедрение ИС

| Наименование | Количество | Стоимость | Сумма |
|---|---|---|---|
| Разработка ИС | 240 часов | 500 руб. | 120 000 руб. |
| Сервер | 1 шт. | 30 000 руб. | 30 000 руб. |
| Внедрение и обучение сотрудников | 80 часов | 350 руб. | 28 000 руб. |
| | | Итого | 178 000 руб. |

На основании приведенных расчетов, полный срок окупаемости капитальных затрат на реализацию проекта составит 12 месяцев.

## Выводы по третьей главе

Выбранная комбинация технологий для разработки ИС обеспечила необходимую функциональную и инструментальную поддержку, достаточное количество готовых библиотек, что в совокупности привело к улучшению качества системы и сокращению трудозатрат на ее разработку. Использование IDEF1X позволило нивелировать различия с логической моделью данных. Применение ORM позволит в дальнейшем выполнить бесшовный переход к любой другой поддерживаемой СУБД. Подсистема Identity позволила обеспечить необходимый уровень разграничения доступа. Примененный паттерн MVC в связке с ASP.NET Core позволили получить гибкую программную архитектуру и упростили процесс тестирования и отладки. Готовый дизайнерский шаблон Inspinia существенно улучшил внешний вид при сохранении удобства использования. Примененный способ построения суррогатной выборки обеспечил высокое качество предсказаний при создании моделей машинного обучения для задачи классификации. Использование в процессе разработки только бесплатно распространяемых компонентов привело к снижению затрат и позволило, в конечном итоге, выйти на экономически привлекательный срок окупаемости проекта.

# Заключение

Развитие и увеличение масштабов деятельности ООО «Проектное бюро Р1» привело к необходимости модернизации бизнес-процессов компании. Так как качество управления финансами напрямую влияет на эффективность работы всего предприятия, то высший менеджмент принял решение провести информатизацию деятельности финансовой службы.

В настоящей работе были рассмотрены деятельность предприятия и финансовой службы, выбрана методология и исследован бизнес-процесс управления платежами. В результате установлено, что устранение выявленных недостатков, посредством разработки и внедрения АИС, приведет к сокращению затрат, повышению скорости и качества работы и, в итоге, улучшит контроль и оперативное управление.

На основании сформированных требований и задач выбрана технология проектирования, с помощью которой выполнены работы по проектированию функционала, ключевых сущностей, состояний, компонентов и модели данных. Применение UML и IDEF1X позволило улучшить качество проектирования, сократило временные затраты и повысило общую продуктивность работы.

Для разработки системы использовался C# и технологическая платформа ASP.NET Core MVC. Встроенная подсистема Identity обеспечила основу для контроля доступа, а EF Core позволила реализовать взаимодействие объектной модели данных с реляционной СУБД.

Для решения задачи классификации оценивалась работа следующих алгоритмов машинного обучения: деревья принятия решения, K-ближайших соседей и логистическая регрессия. В результате проведенных экспериментов удалось получить более 95% точности предсказаний на тестовой выборке.

Проведенная оценка экономической эффективности показала, что внедрение АИС позволит снизить затраты на 70% и окупит расходы на ее разработку и внедрение за 12 месяцев.

## Список используемой литературы и используемых источников


1. Ахметова Г.З. Основы менеджмента : учеб. пособие / Г.З. Ахметова, ОмГТУ. – Омск : Изд-во ОмГТУ, 2019. – 120 с.

2. Белов В.В. Проектирование информационных систем : учебник для студ. учреждений высш. проф. образования / В.В. Белов, В.И. Чистякова ; под ред. В.В. Белова – М.: Издательский центр «Академия», 2013. – 352 с.

3. Блинов А.О. Реинжиниринг бизнес-процессов : учеб. пособие для студентов вузов / А.О. Блинов, О.С. Рудакова, В.Я. Захаров, И.В. Захаров. – М.: ЮНИТИ-ДАНА, 2017. – 344 с.

4. Гвоздева Т.В. Проектирование информационных систем : учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. – Ростов н/Д. : Феникс, 2009. – 508 с.

5. Затонский А.В. Информационные системы : разработки информационных моделей и систем : учеб. пособие / А.В. Затонский. – М.: РИОР: ИНФРА-М, 2020. – 344 с.

6. Заботина Н.Н. Проектирование информационных систем: учеб. пособие. / Н.Н. Заботина. – М.: ИНФРА-М, 2020. – 331 с.

7. Ипатова Э.Р. Методологии и технологии системного проектирования информационных систем [Электронный ресурс] : учебник / Э.Р. Ипатова, Ю.В. Ипатов. – 2-е изд., стер. – М.: ФЛИНТА, 2016. – 256 с.

8. Коваленко В.В. Проектирование информационных систем: учеб. пособие / В.В. Коваленко. – 2-е изд. – Москва: ИНФРА-М, 2021. – 357 с.

9. Кривоносова И.Н. Сравнительный анализ нотаций ARIS и IDEF при описании процессов / И.Н. Кривоносова, Г.Н. Сюткин // Сервис в России и за рубежом. – 2007. – №2. – С. 104-109.

10. Кугаевских А.В. Проектирование информационных систем. Системная и бизнес-аналитика: учеб. пособие / А.В. Кугаевских. – Новосибирск: Изд-во НГТУ, 2018. – 256 с.

11. Назаров С.В. Архитектура и проектирование программных систем : монография / С.В. Назаров. – 2-е изд. – Москва: ИНФРА-М, 2020. – 374 с.

12. Новиков Ф.А. Визуальное конструирование программ / Ф.А. Новиков // Информационно-управляющие системы. – 2005. – №6. – С. 9-22.

13. Норенков И.П. Основы автоматизированного проектирования : учеб. для вузов. 2-е изд. – М.: Изд-во МГТУ им Н.Э. Баумана, 2002. – 336 с.

14. Сервис проверки контрагентов [Электронный ресурс]: Организация ООО «Проектное бюро Р1». URL: https://www.list-org.com/company/7743806 (дата обращения: 25.04.2022).

15. Скородумов П.В. Моделирование бизнес-процессов: подходы, методы, средства // Вопросы территориального развития. – 2014. – №5. – С. 5.

16. Репин В.В. Бизнес-процессы. Моделирование, внедрение, управление / В.В. Репин – М.: Манн, Иванов и Фербер, 2013. – 512 с.

17. Booch G. Object-oriented analysis and design with applications / G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, K. Houston. – 2nd ed. Pearson Education Inc., 2007. – 717 p.

18. Brink H. Real-world machine learning / H. Brink, J. Richards, M. Fetherolf – 1st ed. Manning, 2016. – 264 p.

19. Denis A. System analysis & design: an object-oriented approach with UML / A. Denis, B. Wixom, D. Regarden. – 5th ed. Wiley, 2015. – 544 p.

20. Korobov M. Morphological Analyzer and Generator for Russian and Ukrainian Languages // Proceedings of 3rd Conference on Analysis of Images, Social Networks and Texts (AIST). – 2015. – P. 320–332.

21. Larman C. Applying UML and patterns: an introduction to object-oriented analysis and design and iterative development / C. Larman. – 2rd ed. Addison Wesley Professional, 2004. – 736 p.

22. Maciaszek L.A. Requirements analysis and system design / L.A. Maciaszek – 3rd ed. Pearson Education Limited, 2007. – 651 p.

23. Rumpe B. Modeling with UML: language, concepts, methods / B. Rumpe. – 1st ed. Springer, 2016. – 295 p.

Приложение А

## Список полей данных

Таблица А.1 – Список полей данных

| Модуль | Поля |
|---|---|
| Заявки на оплату | номер заявки (число), дата создания (дата), статус (перечисление), контрагент (справочник), плательщик (справочник), инициатор (справочник), статья затрат (справочник), срок оплаты (дата), договор (текст), счет (текст), сумма (число), описание (текст), остаток (число) |
| Статьи затрат | название (текст), описание (текст), трансферт (логический) |
| Плательщики | название (текст), ИНН (текст), КПП (текст), ОГРН (текст), телефон (текст), адрес (текст) |
| Контрагенты | название (текст), ИНН (текст), КПП (текст), телефон (текст), адрес (текст), e-mail (текст), веб-сайт (текст) |
| Сотрудники | ФИО (текст), телефон (текст), e-mail (текст), должность (текст), офис(текст) |
| Проекты | название (текст), дата начала (текст), дата окончания (текст), ответственный (справочник) |
| Банковские счета плательщиков | расчетный счет (текст), корреспондентский счет (текст), БИК (текст), банк (текст), описание (текст), действующий (логический) |
| Реестр заявок | название (текст), дата создания (дата), дата подготовки (дата), статус (перечисление), ответственный (справочник), дата утверждения (дата), кем согласован (текст) |

Приложение Б

# Форма реестра платежей

| | A | B | C | D | E | F | G | H | I | J | K | L | M | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | **ООО "Проектное бюро Р1"** | | | | | | | | | | | | | | | | | | | |
| 2 | | РЕЕСТР НА ОПЛАТУ | | | | | | 10.02.2022 | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | |
| 4 | | **Банк** | | | | **Входящий** | | **Платежи** | | **Исходящий** | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | |
| 6 | | УБРИР | | | | | | | | | | | | | | | | | | | |
| 7 | | ВТБ | | | | | | | | | | | | | | | | | | | |
| 8 | | АльфаБанк | | | | | | | | | | | | | | | | | | | |
| 9 | | Совкомбанк | | | | | | | | | | | | | | | | | | | |
| 10 | | **ИТОГО** | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | |
| 12 | **№ п/п** | **Статья ДДС** | | | | **Сумма** | | **Проект** | | | | **Контрагент** | | **ИНН** | | **№ договора, дата** | | | **№ счета, дата** | | |
| 13 | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | | |
| 18 | | ИТОГО | | | | | | | | | | | | | | | | | | | |

Рисунок Б.1 – Форма реестра платежей

Приложение В

## Фрагмент кода классификации заявок на оплату

```
full_rows = []
text_rows = []
for row in source_rows:
    db_id = row[0]
    text = clean_text(row[1])
    text = port_stem(text)
    price = minimax(row[2])
    vendor_id = lib_export['vendor_map'].get(row[3], 0)
    project_id = lib_export['project_map'].get(row[4], 0)
    author_id = lib_export['author_map'].get(row[5], 0)
    if (vendor_id != 0 and author_id != 0):
        dict_row = getDummyDict(db_id, text, price, vendor_id, project_id, author_id)
        full_rows.append(dict_row)
    else:
        text_rows.append({'ID': db_id, 'Text': text})

predicted_rows = []

if len(full_rows) > 0:
    pd_full = pd.DataFrame(full_rows)
    pd_full['Price'] = pd_full['Price'].astype(float)
    pd_full_text_vect = cnt_vect.transform(pd_full['Text']).toarray()
    pd_full_text_predict_proba = mnb_model.predict_proba(pd_full_text_vect)
    pd_full_text_proba_df = pd.DataFrame(pd_full_text_predict_proba, index=pd_full.index,
                                         columns=['proba_{}'.format(i + 1) for i in
                                                  range(pd_full_text_predict_proba.shape[1])])
    pd_full_concat = pd.concat([pd_full, pd_full_text_proba_df], axis=1)
    pd_full_concat.loc[:, 'proba_1':] = pd_full_concat.loc[:, 'proba_1':].fillna(1 / 15)
    pd_full_concat['BudgetItem'] = log_model.predict(pd_full_concat.drop(['ID', 'Text'], axis=1).values)
    for index, row in pd_full_concat.iterrows():
        predicted_rows.append({'ID': row['ID'], 'BudgetItem': row['BudgetItem']})

if len(text_rows) > 0:
    pd_textonly = pd.DataFrame(text_rows)
    pd_textonly_text_vect = cnt_vect.transform(pd_textonly['Text']).toarray()
    pd_textonly['BudgetItem'] = mnb_model.predict(pd_textonly_text_vect)
    for index, row in pd_textonly.iterrows():
        predicted_rows.append({'ID': row['ID'], 'BudgetItem': row['BudgetItem']})

if len(predicted_rows) > 0:
    connection = pyodbc.connect(connString)
    cursor = connection.cursor()
    for row in predicted_rows:
        cursor.execute("UPDATE PaymentRequests SET RegisterCodeId=? where Id=?", row['BudgetItem'], row['ID'])
        connection.commit()

    cursor.close()
    connection.close()

print("ML Predict completed")
```

Рисунок В.1 – Фрагмент кода классификации заявок на оплату