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

Кафедра «Прикладная математика и информатика
(наименование)

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

Цифровая трансформация бизнеса
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: Разработка проекта автоматизации процесса анализа показателей медицинских организаций с использованием BI-системы

Обучающийся Н. А. Нижалов _____
(Инициалы Фамилия) (личная подпись)

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

Консультант И.Ю. Усатова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы «Разработка проекта автоматизации процесса анализа показателей медицинских организаций с использованием BI-системы».

Целью ВКР является автоматизация процесса анализа показателей медицинских организаций.

Основные задачи: проведение анализа текущих процессов ручной обработки данных, разработка функциональной модели автоматизированной системы, создание программного обеспечения для визуализации данных, оценка экономической эффективности проекта.

Предмет исследования - процессы обработки данных о доступности электронных записей в медицинских учреждениях и их автоматизация.

Выпускная квалификационная работа состоит из введения, трех глав и заключения.

Глава 1 посвящена функциональному моделированию предметной области, включая анализ текущих процессов (AS-IS) и проектирование улучшенной модели (TO-BE).

Глава 2 описывает выбор архитектуры автоматизированной информационной системы (АИС), логическое моделирование и требования к аппаратно-программному обеспечению.

Глава 3 охватывает физическое проектирование АИС, разработку программного обеспечения, описание функциональности и оценку экономической эффективности.

Заключение: разработанная система успешно автоматизирует обработку данных, сокращает временные затраты и минимизирует ошибки, связанные с ручной обработкой. Проект демонстрирует высокую экономическую эффективность и готов к внедрению.

Общий объем работы: 73 страницы.

Abstract

The title of the graduation work is "Development of a Project for Automating the Analysis of Medical Organization Indicators Using a BI System".

The graduation work consists of an introduction, three parts, 22 figures, a conclusion, and a list of 25 references including foreign sources.

The aim of this graduation work is to automate the process of analyzing medical organization indicators, focusing on the availability of electronic patient records.

The object of the graduation work is the data processing workflows in medical institutions.

The subject of the graduation work is the automation of these processes using a BI system.

The key issue of the graduation work is the inefficiency of manual data processing, which is time-consuming and prone to errors.

The graduation work may be divided into several logically connected parts:

- The first part describes the functional modeling of the subject area, including the analysis of current processes (AS-IS) and the design of an improved model (TO-BE).
- The second part outlines the architecture of the automated information system (AIS), logical modeling, and hardware/software requirements.
- The third part covers the physical design of the AIS, software development, functionality description, and economic efficiency evaluation.

In conclusion, the developed system successfully automates data processing, significantly reducing time costs and minimizing errors associated with manual handling. The project demonstrates high economic efficiency and is ready for implementation.

The work is of interest for a wide circle of readers involved in healthcare IT, data analysis, and process automation

Оглавление

Введение

Современные медицинские учреждения сталкиваются с необходимостью оперативного анализа данных для принятия управленческих решений. Существует проблема ручной обработки данных, связанных с контролем доступности электронной записи пациентов. Это требует значительных временных затрат и может приводить к ошибкам.

Задача ВКР: автоматизация процесса анализа показателей медицинских организаций, включая подготовку отчетов для еженедельных видеоконференций с Министерством здравоохранения Вологодской области.

Автоматизация процессов анализа данных в здравоохранении является актуальной задачей, так как:

- ручная обработка данных занимает много времени и подвержена ошибкам;
- оперативное получение аналитики необходимо для принятия управленческих решений;
- существующие решения требуют значительных финансовых затрат и времени на внедрение.

Разработанная система позволит сократить временные затраты, минимизировать ошибки и повысить эффективность управления ресурсами здравоохранения.

Целью ВКР является разработка проекта автоматизации процесса анализа показателей медицинских организаций с использованием BI-системы.

Основные задачи:

- провести анализ текущих процессов ручной обработки данных;
- разработать функциональную модель автоматизированной системы;
- создать программное обеспечение для визуализации данных;
- оценить экономическую эффективность проекта.

Объект исследования - процесс обработки данных о доступности электронных записей в медицинских учреждениях.

Предмет исследования - автоматизация процесса анализа показателей медицинских организаций процессов с использованием BI-системы.

Для реализации проекта использованы следующие методы:

- функциональное моделирование (модели AS-IS и TO-BE) для анализа и оптимизации процессов;
- UML для проектирования архитектуры системы;
- разработка ПО на Python с использованием библиотек для обработки данных (xlrd) и визуализации (Chart.js);
- экономический анализ для оценки эффективности внедрения.

Выбор этих методов обусловлен их эффективностью для решения поставленных задач и соответствием современным стандартам разработки.

Разработанная система:

- сокращает время подготовки отчетов с 8 часов до нескольких минут;
- минимизирует ошибки, связанные с ручной обработкой;
- позволяет оперативно анализировать данные для принятия управленческих решений;
- экономит бюджетные средства (до 10 млн рублей по сравнению с закупкой готового решения).

Глава 1 Функциональное моделирование предметной области

1.1 Сфера деятельности организации

Разработка проекта проходила в Бюджетном Учреждении Вологодской области «Медицинские цифровые технологии». Основными видами деятельности организации является – обеспечение функционирования информационных систем, а именно внедрение программного обеспечения для автоматизированных систем сбора, обработки, хранения и передачи информации в сфере здравоохранения области, совершенствование регионального сегмента Единой государственной информационной системы в сфере здравоохранения Вологодской области, эффективное и бесперебойное функционирование информационных систем, их техническая поддержка и сопровождение, включая централизованное формирование региональных справочников, интеграция иных информационных систем, отвечающих техническим требованиям к информационным системам в сфере здравоохранения.

1.2 Краткая характеристика деятельности организации

Бюджетное учреждение Вологодской области «Медицинские Цифровые Технологии» является молодой организацией в сфере ИТ-услуг и существует всего несколько лет. Организационно-правовая форма – Государственные бюджетные учреждения субъектов Российской Федерации, где права учредителя осуществляет Департамент цифрового развития области. В настоящий момент учреждение работает с медицинскими организациями области по внедрению и сопровождению информационных систем Вологодской области, таких как ИС «РМИС ВО», ИС «Информационная лабораторная система» (ЛИС), ИС «Центральный архив медицинских изображений» (ЦАМИ), ИС «Скорой медицинской помощи» (СМП), ИС

«Телемедицинские консультации Вологодской области» (ТМК). Миссия БУ «МЦТ» внедрение современных информационных технологий для повышения качества медицинской помощи и эффективности использования ресурсов здравоохранения. Структура организации (рисунок 1) делится на техническое и административное направление.

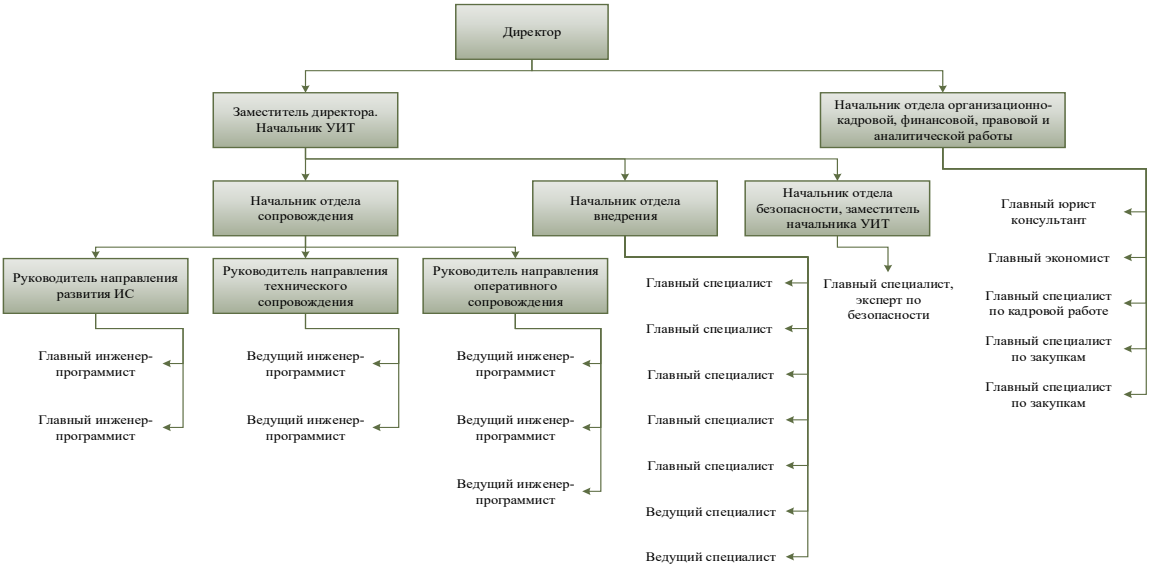

Рисунок 1 – Организационная структура учреждения

В рамках административного направления ведется основной документооборот по кадровой части, оформлении документов по закупкам и решению юридических вопрос.

Техническое направление делится на отдел сопровождения и отдел внедрения. В отделе сопровождения оказывается консультативная помощь по работе в системах, а в отделе внедрения подготавливаются контракты на развитие системы.

1.3 Характеристика подразделения организации

Работа над проектом проходила в отделе сопровождение медицинских информационных систем. На сегодняшний день медицинская организация,

уже использующая РМИС, в существенной степени зависит от качества системы и, как следствие, качества ее сопровождения. Пользователи медицинских организаций не просто используют РМИС как какую-то программу для разового внесения данных, которые нужно отдать кому-то «наверх» или как обязанность, которую их заставляют выполнять. В целом сопровождение МИС представляет собой организацию и постоянное совершенствование работы службы Service Desk (служба сопровождения). Нередко такой сервис по привычке называют «службой технической поддержки», хотя данный термин является на самом деле недостаточным для медицинских информационных систем, так как речь идет не только и не столько о технической поддержке, сколько о полноценном сопровождении всей системы. В тех вопросах, в которых собственные специалисты внутренней ИТ-службы медицинской организации не смогли разобраться сами или когда им требуется контакт с разработчиком, они обращаются в соответствующую службу. В структуре учреждения помимо отдела сопровождения функционирует отдел внедрения медицинских информационных систем. Основной задачей, которого является подготовка объекта к внедрению, опытное внедрение, сдача его в промышленную эксплуатацию

В отделе сопровождения основная работа ведется через систему «Помощник». В которой медицинские организации создают заявки с конкретной проблемой. После создания заявки она попадает на диспетчеров, которые проводят первичный анализ заявки, выставляют ей уровень сложности и выбирают ответственного сотрудника за выполнение данной заявки. Ответственный сотрудник берет заявку в работу, прорабатывает вопрос и предоставляет по ней ответ. При проработке заявки специалист использует доступные инструкции, тестовую базу для воспроизведения проблемы. Так же в отделе сопровождения выполняются поручения от руководства по подготовке отчетов и сбору статистической информации по медицинской организации или по всему региону в целом.

1.4 Описание предметно-ориентированных информационных систем

В организации используется 6 предметно-ориентированные системы: Directum RX, региональная медицинская информационная система, ИС «Скорая медицинская помощь», ИС «Телемедицинские консультации Вологодской области», ИС «Центральный архив медицинских изображений», ИС «Мониторинг беременных Вологодской области»:

- система электронного документооборота и управления взаимодействием «Directum RX» – программа используется для постановки, отслеживания и принятия задач в рамках отделов, так же используется для документооборота входящих и исходящих писем, запросов и обращений. Удобная программа для взаимодействия с документами и отслеживания задач;
- ИС «РМИС – региональная медицинская информационная система включает в себя работу как с электронной медицинской картой, в которой отслеживаются все случаи лечения, связанные с поликлиникой, стационаром, реанимацией и скорой медицинской помощи, так же в ней хранятся все анализы, которые делаются в лаборатории. Медицинские организации настраивают в системе структуру, штатное расписание и работающих врачей, дальше в соответствии с должностью врачи получают доступ к рабочим местам в программе (поликлиника, стационар, лаборатория, диагностика). Данная система так же имеет интеграцию с другими системами, такими как ИС «СМП» (выгружаются карты вызова), ИС «ЦАМИ» (получает доступ к снимкам), ИС «МБ ВО» (передаются данные по отслеживанию случаев беременности). В данной системе создаются электронные медицинские документы, которые направляются в региональный и федеральный реестр электронных медицинских документов, после чего вся информация

по случаям лечения и документам отображается в личном кабинете пациента на портале ЕПГУ (включая электронные больничные). Плюсом системы является то, что большой объем информации находится в одном место то, что врачам удобно узнать историю болезни пациента, не делая запросов в другие медицинские организации и не тратя время. Минус в том, что при обновлениях системы могут возникать проблемы, которые уже находят медицинские организации (из-за большого функционала, тестировщики не могут проверить весь объем программы, потому что есть множество взаимодействий во всей системе в целом). Так же из-за большого количества изменений в документах, на основании которых работают врачи, система довольно часто обновляется;

– ИС «СМП» – скорая медицинская помощь включает в себя отслеживание машин скорой помощи системой «Глонасс», ведение смен врачей, выезжающих на выезды, занесение информации о выезде. В медицинских организациях сидит диспетчер, который принимает вызов и передает в работу бригаде. Бригада направляется на вызов, отрабатывает его и параллельно или после вызова фиксирует всю информацию в карте вызова через планшет, который через интернет по защищенным канал связи передает информацию на сервер, если планшет будет без интернета, то информация сохраняется в архив и при доступе к интернету передает информацию на сервер. Как только карта вызова будет подписана главным фельдшером, она передается в РМИС и вся информация фиксируется в электронной медицинской карте пациента. Плюсами является то что диспетчер может отслеживать где находится бригада, быть всегда с ней на связи и в случае возникновения каких либо проблем может связаться с данной бригадой или вызвать необходимую помощь;

– ИС «ТМК ВО» – телемедицинские консультации, при болезни, не требующей посещения поликлиники или в каких-либо ещё случаях пациент может записаться ко врачу на онлайн – консультацию. У врача и пациента есть возможно поговорить по аудио и видео связи. Врач может обсудить или при необходимости увидеть проблему пациента и проконсультировать его по ней. Большой плюс системы заключается в том, что консультация проводится без вероятности заражения болезнью, которая передается при общении, так же экономит большое количество времени как пациента, так и врача;
– ИС «ЦАМИ» – центральный архив медицинских изображений, данная система работает в связке с РМИС, когда пациенту делается запрос на диагностику (к примеру рентген) из РМИС уходит направление в ИС «ЦАМИ» и фиксируется созданное направление. Пациент приходит на прием, лаборант делает снимок и связывает его с направлением, после чего врач пишет результат и подписывает (что тоже является электронным медицинским документом). Далее пациент приходит к терапевту, и врач уже через РМИС может увидеть связанный с протоколом снимок, который находится в ИС «ЦАМИ». Плюс в том, что можно сделать покадровое видео, посмотреть его по кадрам, так же нанести необходимые пометки (как на снимок, так и на серию снимков);
– ИС «МБ ВО» – мониторинг беременных, система для проведения полного жизненного цикла беременных пациенток, фиксируются от первичных обращения, до нахождения в стационаре во время беременности. После родов пациентки так же наблюдается в течении определенного времени. В системе проводятся тесты на возможные патологии у новорожденных. Информация из системы так же передается в РМИС. Из-за всей специфики имеющей во время беременности была выведенная отдельная система.

1.5 Функциональная модель организации «AS-IS»

Модель AS-IS («Как есть») – это модель для описания бизнес-процессов в организации [15]. С её помощью фиксируют и визуализируют, как организованы процессы прямо сейчас, в данный момент.

Каждую неделю в регионе происходя ВКС (видеоконференции) в которых принимают участие Министерство здравоохранения Вологодской области, медицинские организации Вологодской области и медицинские цифровые технологии. На данных ВКС разбираются острые вопросы региона, анализ выполнения поставленных показателей Минздравом.

К каждой встрече подготавливается информация, статистика по выполнению показателей в разрезе каждой медицинской организации и общая по всему региону.

Одним из важных показателей является возможность записи и фактическая запись на прием к врачу через ЕПГУ.

Построим модель AS-IS по подготовке визуализированных данных к ВКС, модель показана на рисунке 2.

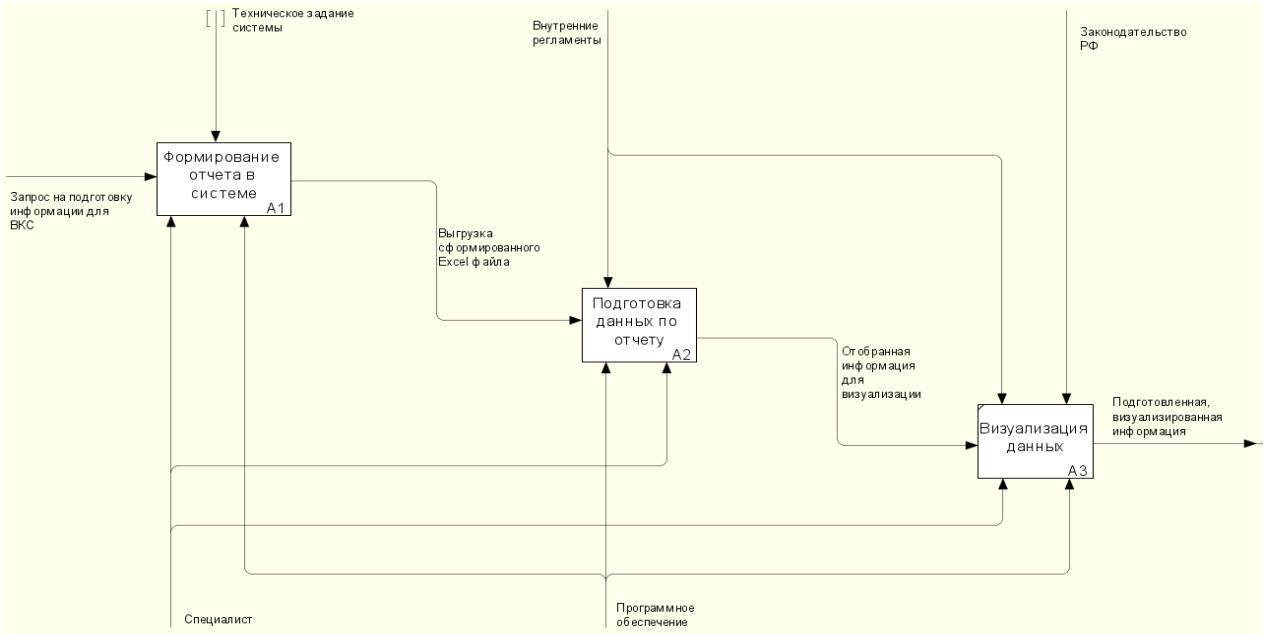

Рисунок 2 – Подготовка визуализированных данных к ВКС

Для формирования отчета в системе специалисту необходимо выбрать отчет в системе РМИС, указать необходимый период формирования и нажать кнопку сформировать отчет. После нажатия кнопки отчет будет загружен на компьютер. Декомпозиция блока А1 функциональной модели «Формирование отчета в системе» показывает, как устроена работа на данный момент, представлена на рисунке 3.

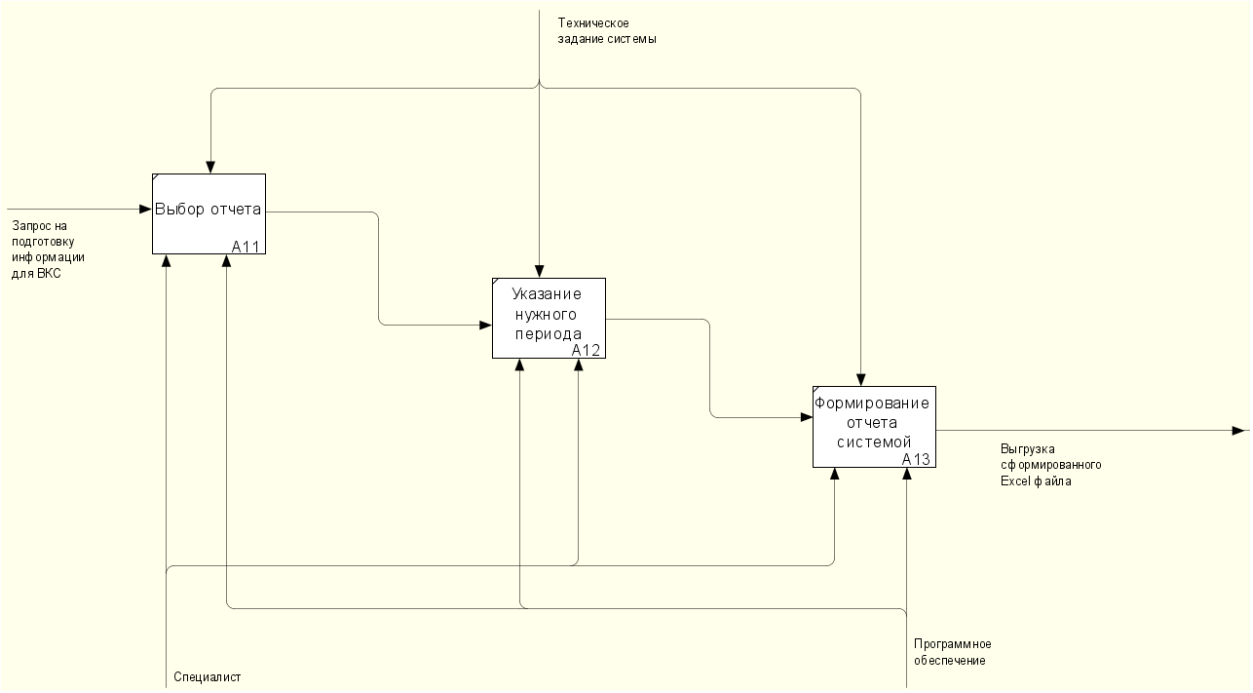

Рисунок 3 – Декомпозиция блока А1 функциональной модели «Формирование отчета в системе»

После выгрузки отчета специалист переходит к подготовке данных к ВКС, т.к. из системы выгружается не готовая информация, которая предоставляется на ВКС, а только собирается статистика за период, из-за чего специалисту приходится ее обрабатывать. В рамках подготовки специалист удаляет лишнюю информацию из отчета, производит сортировку оставшихся данных и рассчитывает для каждой медицинской организации плановые показатели для записи через ЕПГУ (составляет не менее 60% от всех созданных записей). Плановые показатели по количеству занятых бирок пациентами через ЕПГУ устанавливаются приказом МЗО на регион.

Декомпозиция блока А2 функциональной модели «Подготовка данных по отчету» показывает, как устроена работа на данный момент, декомпозиция представлена на рисунке 4.

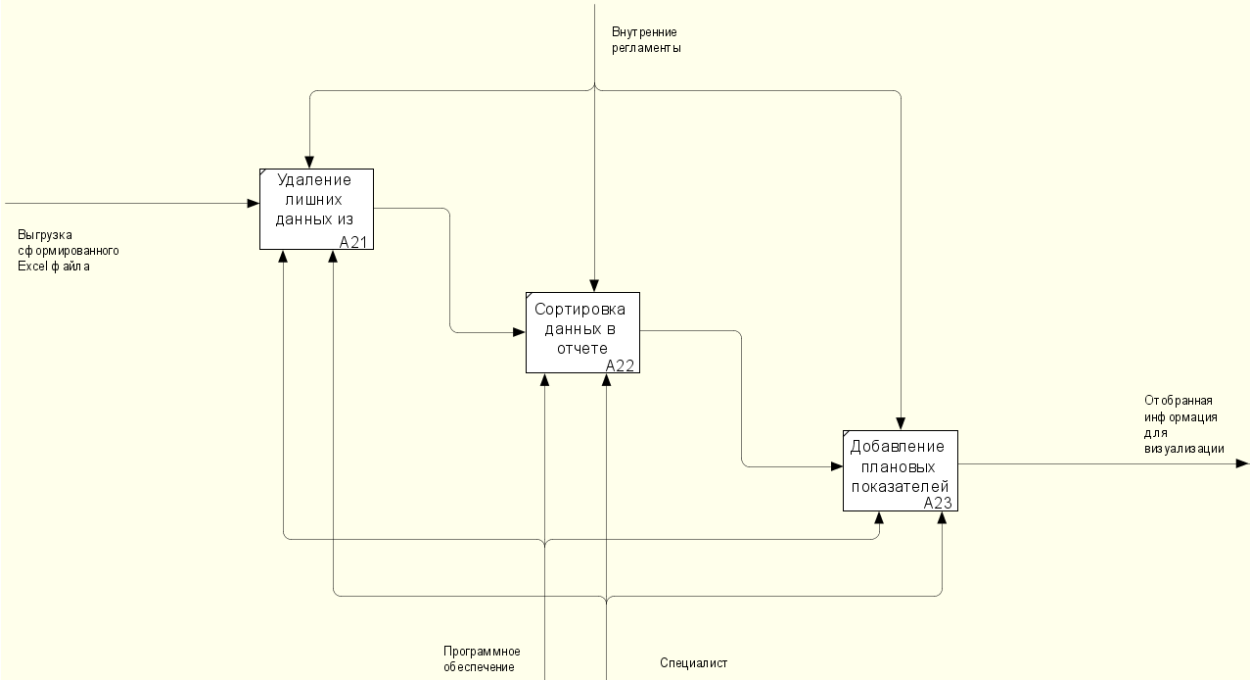

Рисунок 4 – Декомпозиция блока А2 функциональной модели «Подготовка данных по отчету»

После подготовки данных отчета специалист их визуализирует для предоставления информации на ВКС.

1.6 Функциональная модель организации «ТО-ВЕ»

Модель ТО-ВЕ («Как должно быть») – это одна из моделей для описания бизнес-процессов [15]. Она помогает описать идеальное или улучшенное состояние организации и работы всех её подразделений [19, 20].

В рамках модели AS-IS было выявлено, что тратиться большое количество времени на подготовку отчета и его визуализацию. Информация по показателю проверяется и озвучивается каждую неделю, следовательно тратиться время на подготовку данной информации так же еженедельно.

Для оптимизации работы по обработке и визуализации данных было принято решение о разработке отдельной программы, которая автоматизирует данный процесс.

В модели TO-BE будут процессы выгрузка отчета из РМИС, автоматизированный анализ данных в BI-системе и визуализация отчета, данная модель показана на рисунке 5.

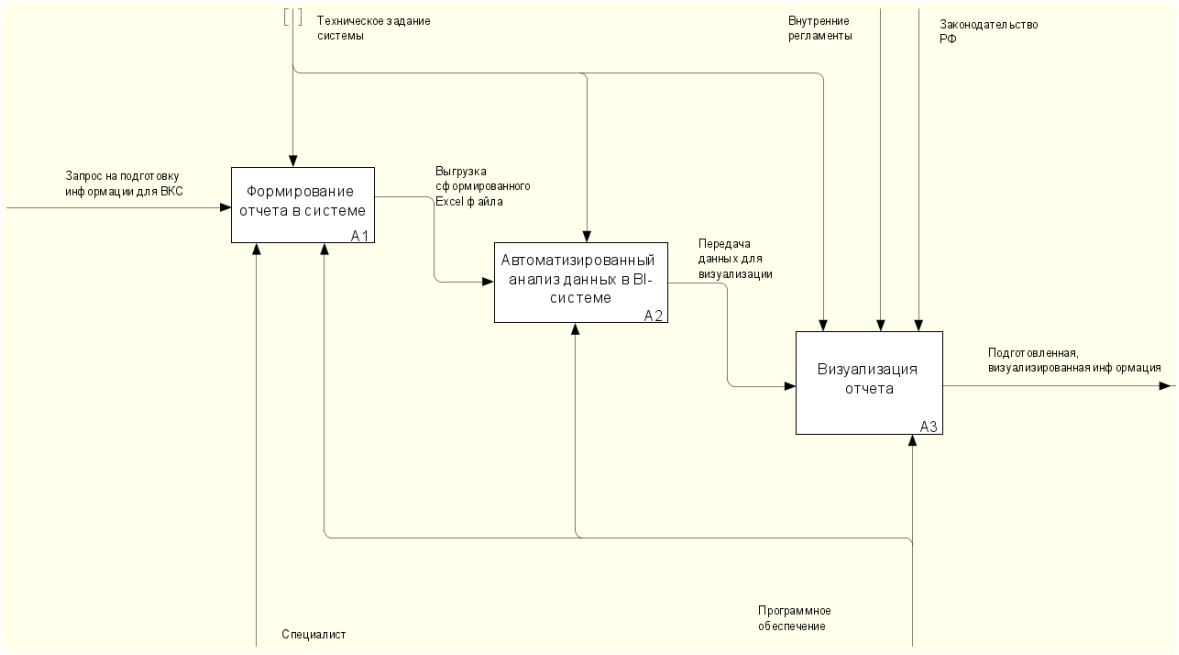

Рисунок 5 – Подготовка визуализированных данных к ВКС в модели TO-BE

Декомпозиция блока A1 функциональной модели «Формирование отчета в системе» сохранила свою структуру из модели AS-IS, поскольку данный этап не требовал изменений и включает в себя стандартную процедуру выгрузки данных из РМИС.

Декомпозиция блока отражает принципиальные изменения в процессе обработки данных. После выгрузки отчет автоматически поступает в BI-систему, где выполняются расчеты ключевых показателей, включая процент записей через ЕПГУ, с последующей детализацией по ЛПУ. Основное отличие новой модели заключается в полной автоматизации процессов обработки и анализа данных, что исключает ручные операции и связанные с ними ошибки,

значительно ускоряя подготовку информации для визуализации. На схеме четко отображен усовершенствованный процесс обработки, в рамках которого информация последовательно проходит этапы автоматизированного анализа, контроля качества и приведения к единому формату для последующего графического представления. Декомпозиция блока А2 функциональной модели «Автоматизированный анализ данных в BI-системе», представлена на рисунке 6.

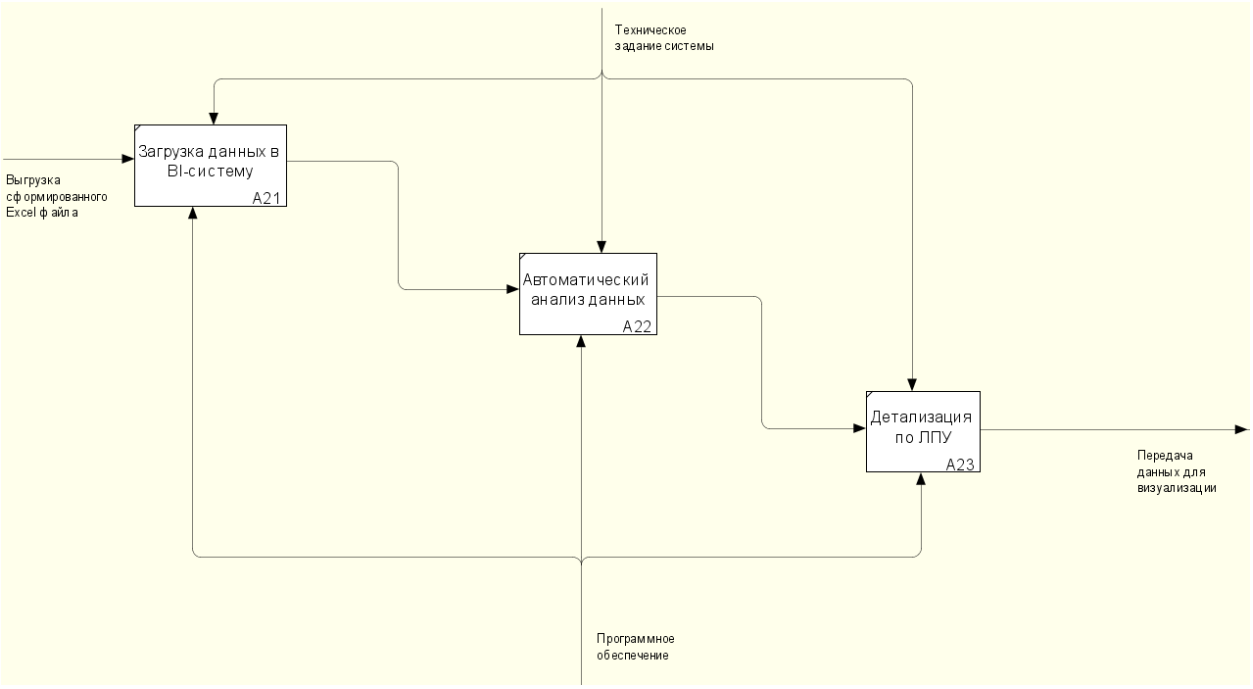

Рисунок 6 – Декомпозиция блока А2 функциональной модели «Автоматизированный анализ данных в BI-системе»

На схеме представлен завершающий этап обработки данных - их графическое представление в виде интерактивного отчета. После выполнения всех расчетов и аналитических процедур система автоматически формирует виртуальную среду для отображения информации, где данные структурируются в удобном для восприятия виде. Особенностью данного процесса является динамическое построение информационной панели, которая наглядно отображает ключевые показатели по записи пациентов через

ЕПГУ с возможностью детализации по отдельным медицинским учреждениям.

Сформированная информационная панель обеспечивает комплексное представление данных, включая не только текущие показатели доступности электронной записи, но и инструменты для их динамического сравнения во временном разрезе. Особое внимание уделено адаптации визуализации под специфику разных категорий медучреждений - центральных районных больниц, поликлиник и специализированных диспансеров, что позволяет получать детализированную аналитику для каждого типа организаций. Декомпозиция блока А2 функциональной модели «Визуализация отчета через программу», представлена на рисунке 7.

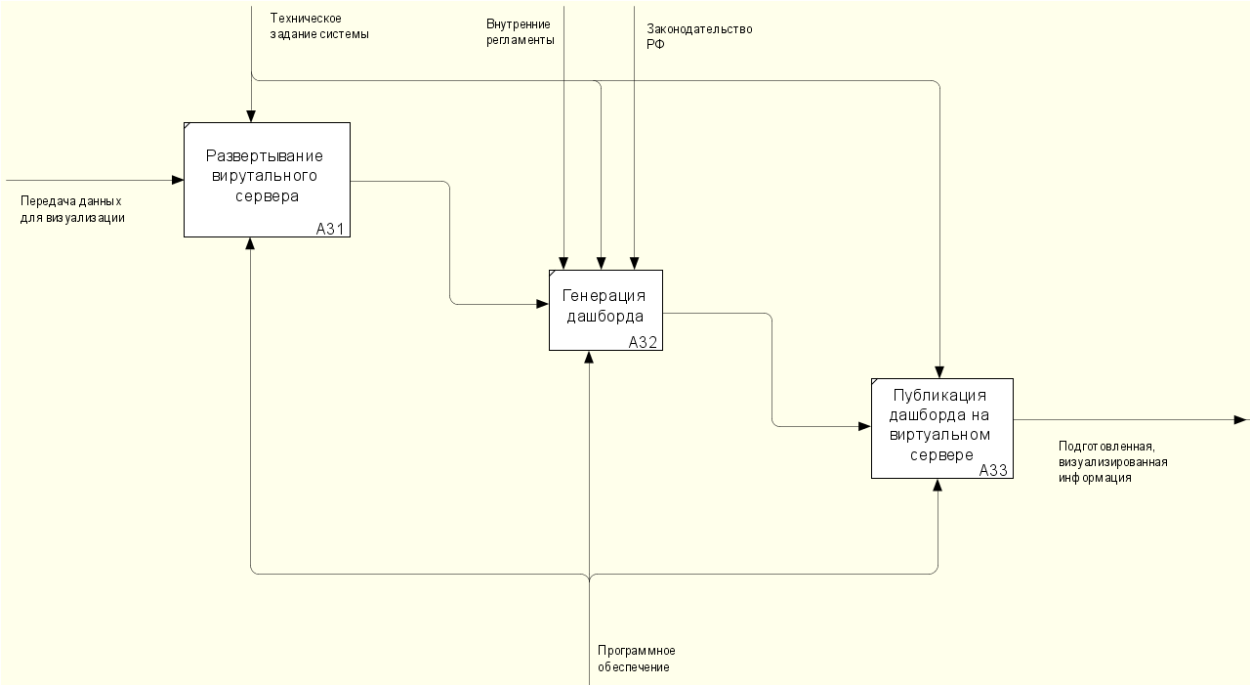

Рисунок 7 – Декомпозиция блока А2 функциональной модели «Визуализация отчета через программу»

В результате анализа планируется реализовать программу, с помощью которой можно анализировать информацию по выполнению показателей медицинских организаций по созданию бирок для записи через ЕПГУ и фактическую запись пациентов на них в рамках всего региона.

1.7 Анализ существующих разработок

На текущий момент существует только одна разработка по решению задачи анализа и визуализации данных. Данный функционал уже реализован у разработчиков системы РМИС, но для возможности его использования необходимо включить в контракт на развитие данный функционал и после закупить на регион. Исходя из этого самостоятельная разработка данного функционал экономит время на включение в контракт развития системы, согласования его с департаментом здравоохранения Вологодской области, заключения контракта с разработчиками и добавления функционала на регион. По предварительным оценкам данный функционал экономит около 6 месяцев. Так же у разработчиков было запрошено коммерческое предложение по данному функционалу и исходя из него будет сэкономлено около 10 миллионов рублей регионального бюджета.

1.8 Описание функциональных требований проекта

В рамках разработанной программы необходима возможность визуализации данных по возможности записи пациентам в медицинскую организацию через ЕПГУ. Информация должна отображаться по количеству созданных записей, сколько должно быть доступных записей через ЕПГУ, сколько доступно записей через ЕПГУ и сколько было записано пациентов через ЕПГУ. Информация по записи должна быть разбита на ЦРБ, поликлиники и диспансеров/специальные МО.

Программа визуализирует информацию по скаченному из системы РМИС отчету, для удобства ее анализа.

FURPS – классификация требований к программным системам.

Образована от первых букв слов:

– Functionality – функциональные требования по ГОСТ 34.602-89: свойства, возможности, безопасность. Являются основными, по

этим требованиям строятся диаграммы вариантов использования (Use case diagram);

- Usability – требования к удобству использования (UX): человеческий фактор, эстетика, последовательность, документация;
- Reliability – требования к надёжности: частота возможных сбоев, отказоустойчивость, восстанавливаемость, предсказуемость устойчивости;
- Performance – требования к производительности: время отклика, использование ресурсов, эффективность, мощность, масштабируемость;
- Supportability – Требования к поддержке: возможность поддержки, ремонтопригодность, гибкость, модифицируемость, модульность, расширяемость, возможность локализации.

Функциональные требования.

Ввод данных**:** система должна принимать данные из внешнего файла Excel в заранее определенном формате. Формат включает столбцы с названием ЛПУ, специальностью, общим количеством слотов для ЕПГУ и ЛПУ, а также количеством занятых слотов для ЕПГУ и ЛПУ. Необходимо обрабатывать ошибки: файл не найден, некорректный формат данных.

Обработка данных**:** система должна обрабатывать входные данные для расчета:

- общего количества слотов для каждого ЛПУ;
- общего количества слотов ЕПГУ для каждого ЛПУ;
- количества занятых слотов ЕПГУ для каждого ЛПУ.

Агрегация данных**:** система должна агрегировать данные для создания трех наборов данных для каждой группы ЛПУ:

- общее количество слотов;
- количество слотов ЕПГУ;
- количество занятых слотов ЕПГУ.

Визуализация данных**:** система должна генерировать и отображать гистограммы. Каждая гистограмма должна отображать агрегированные данные для определенной группы ЛПУ. Гистограммы должны иметь четкие подписи осей (названия ЛПУ и количество слотов) и подписи значений на каждом столбце.

Не функциональные требования.

Требования к удобству использования (Usability):

Интуитивный интерфейс: пользовательский интерфейс должен быть простым и понятным для навигации. Выбор файла отчета должен быть простым и очевидным.

Ясное представление данных: созданные гистограммы должны быть легко интерпретируемыми, с четкими подписями и легендами.

Обработка ошибок: система должна выдавать информативные сообщения об ошибках, если файл не найден или данные некорректны.

Быстродействие: время отклика системы на загрузку данных и построение графиков должно быть минимальным.

Требования к надежности (Reliability):

целостность данных: система должна гарантировать точное считывание данных из файла Excel и их корректную обработку.

Обработка ошибок: надежная обработка ошибок необходима для предотвращения сбоев и повреждения данных.

Проверка данных: система должна проверять входные данные для обеспечения согласованности и точности. Проверка корректности входных данных критически важна.

Стабильность системы: система должна быть стабильной и надежной даже при большом объеме входных данных.

Требования к производительности (Performance):

время загрузки данных: время, затрачиваемое на загрузку и обработку данных Excel, должно быть минимальным, особенно для больших файлов.

Время построения графиков: время, необходимое для построения

гистограмм, должно быть быстрым.

Потребление ресурсов: система должна эффективно использовать системные ресурсы (CPU, память).

Требования к поддерживаемости (Supportability):

обслуживаемость: код должен быть хорошо структурированным, документированным и легко поддерживаемым и обновляемым.

Расширяемость: архитектура системы должна допускать будущее расширение и добавление новых функций.

Тестируемость: код должен быть разработан так, чтобы его легко можно было тестировать.

Для наглядного представления взаимодействия пользователей с системой разработана диаграмма вариантов использования (рисунок 8). На ней отражены ключевые роли и функции, описанные в функциональных требованиях (FURPS).

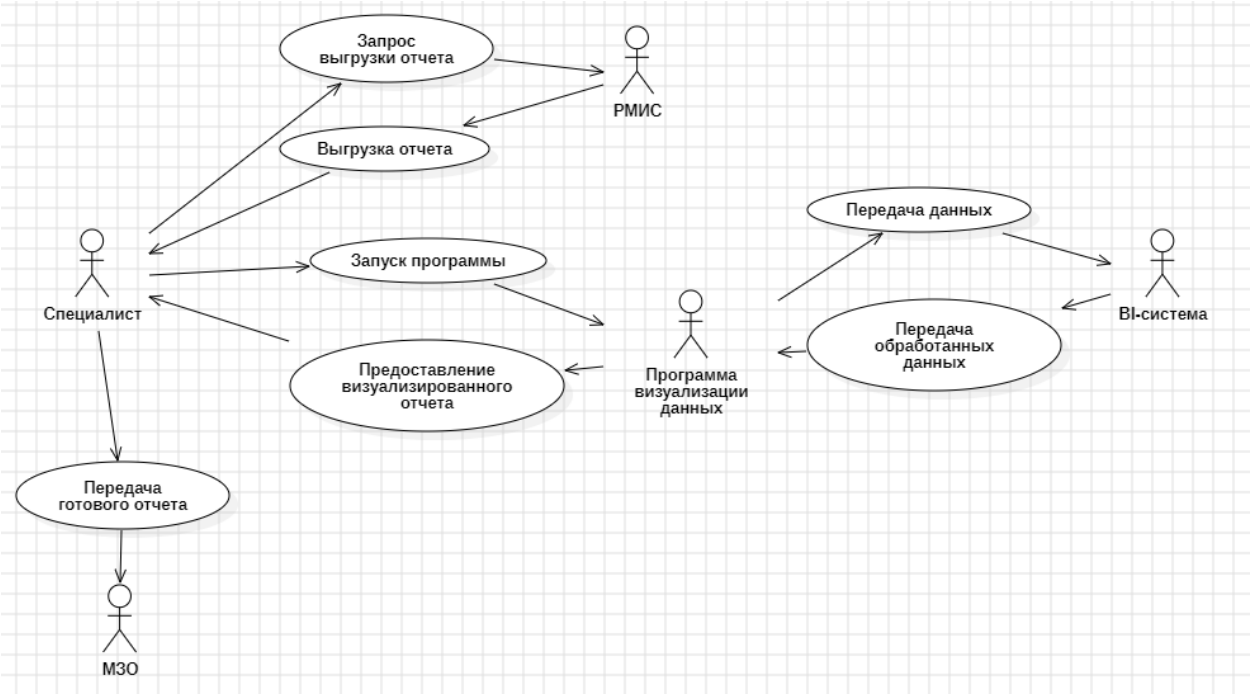

Рисунок 8 – Диаграмма вариантов использования

Система автоматизирует процессы обработки данных, исключая ручные операции.

Вывод по главе

В первой главе проведен анализ деятельности бюджетного учреждения Вологодской области «Медицинские цифровые технологии» (БУ ВО «МЦТ»), которое занимается внедрением и сопровождением информационных систем в сфере здравоохранения. Основное внимание уделено отделу сопровождения медицинских информационных систем, где осуществляется поддержка и совершенствование работы системы «РМИС» (региональная медицинская информационная система).

В рамках анализа выявлены ключевые бизнес-процессы, связанные с подготовкой данных для видеоконференций (ВКС) с участием Министерства здравоохранения Вологодской области. Была построена функциональная модель AS-IS, которая отражает текущее состояние процессов, включая ручную обработку данных и формирование отчетов. На основе этой модели выявлены проблемы, связанные с большими временными затратами на подготовку и визуализацию данных.

Для оптимизации процессов предложена функциональная модель TO-BE, которая предполагает автоматизацию обработки данных и их визуализацию с помощью разработанной программы. Это позволит значительно сократить время на подготовку отчетов и минимизировать ошибки, связанные с ручной обработкой данных.

Также проведен анализ существующих разработок и предложено решение, которое экономит время и бюджетные средства по сравнению с закупкой аналогичного функционала у сторонних разработчиков. Описаны функциональные требования к системе, которые включают ввод данных, их обработку, агрегацию и визуализацию.

В целом, первая глава демонстрирует необходимость автоматизации процессов обработки данных и обосновывает разработку программного решения, которое повысит эффективность работы и сократит затраты на подготовку отчетов.

Глава 2 Выбор архитектуры автоматизированной информационной системы

2.1 Выбор технологии логического моделирования АИС

Для проектирования логической структуры данных и процессов в системе была выбрана технология UML (Unified Modeling Language) [3]. Этот инструмент был выбран благодаря его удобству, широким возможностям для создания профессиональных диаграмм [6].

С помощью UML была разработана диаграмма классов, которая отображает основные сущности системы и их взаимосвязи представлены на рисунке 9.

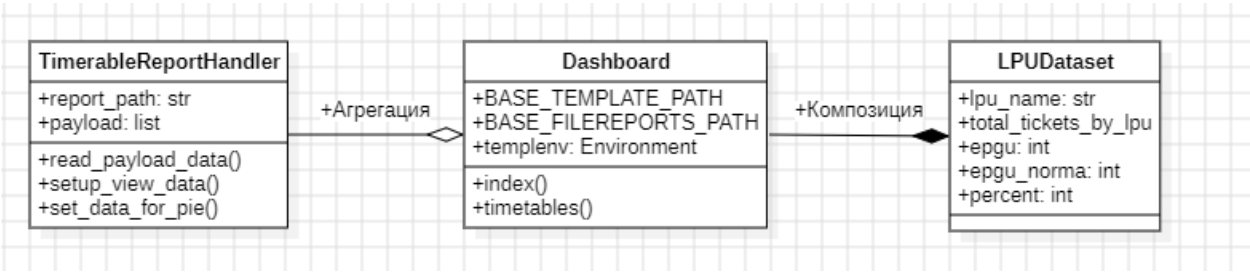

Рисунок 9 – Диаграмма классов

На диаграмме представлены следующие классы:

- TimetableReportHandler – класс, отвечающий за обработку данных из файлов Excel. Он содержит методы для чтения данных, их обработки и подготовки к визуализации;
- Dashboard – класс, реализующий веб-приложение для отображения статистики. Он взаимодействует с классом TimetableReportHandler для получения данных и отображает их через HTML-шаблоны;
- LPUDataset – класс, представляющий набор данных по лечебно-профилактическим учреждениям (ЛПУ). Он содержит поля для

хранения названия ЛПУ, количества слотов, занятых слотов и других статистических данных.

Также была разработана диаграмма последовательностей, которая описывает процесс обработки данных в системе, она представлена на рисунке 10.

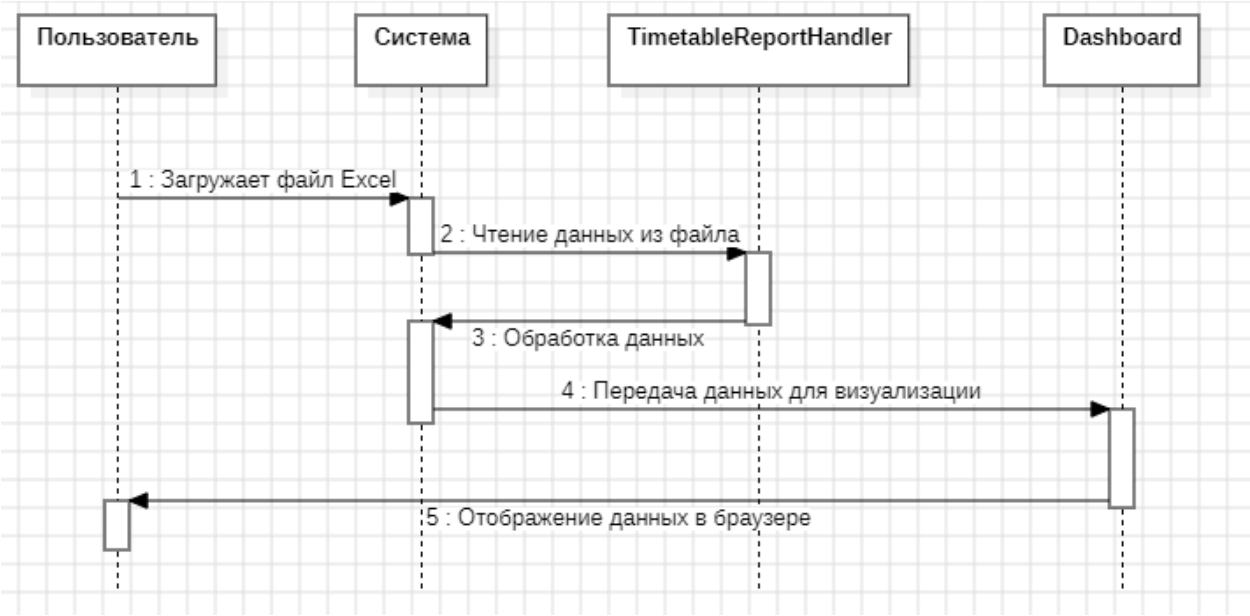

Рисунок 10 – Диаграмма последовательностей процесса обработки и визуализации данных из Excel-файла в веб-приложении

На диаграмме отображены следующие шаги:

- пользователь загружает файл Excel с данными о расписании;
- система с помощью класса TimetableReportHandler читает данные из файла;
- данные обрабатываются и преобразуются в объекты Python;
- класс Dashboard получает обработанные данные и передает их в HTML-шаблон для визуализации;
- пользователь видит результаты визуализации в веб-браузере.

Диаграмма последовательностей была создана с использованием элементов UML, таких как объекты, сообщения и временные линии. Это позволило четко описать процесс взаимодействия между компонентами

системы и пользователем.

Логическое моделирование системы выполнялось поэтапно. На первом этапе проведен анализ предметной области: изучены процессы ручной обработки отчетов в БУ ВО «МЦТ», выделены ключевые сущности («ЛПУ», «Слоты записи», «Отчеты») и их атрибуты. Для проектирования использован стандарт UML 2.5, обеспечивающий наглядное представление структуры данных и процессов. Валидация модели выполнена на соответствие требованиям ФЗ-323 «Об основах охраны здоровья» и внутренним регламентам учреждения.

В качестве инструментария применен Visual Paradigm 17.0, позволяющий генерировать диаграммы в стандартизированном формате. Выбор технологии UML обоснован ее совместимостью с ГОСТ 34.201-89 и возможностью интеграции с другими этапами проектирования АИС. Модель включает диаграммы классов, последовательностей и состояний, охватывающие все аспекты работы системы.

2.2 Логическая модель АИС и ее описание

Логическая модель системы описывает структуру данных и их взаимосвязи по ГОСТ 34.201-89. В данном проекте данные хранятся в файлах Excel, поэтому логическая модель была построена на основе структуры этих файлов.

Структура данных, файл Excel:

- название ЛПУ (лечебно-профилактическое учреждение) – текстовое поле;
- должность (например, «врач-хирург») – текстовое поле;
- количество слотов для ЕПГУ (единый портал государственных услуг) – числовое поле;
- количество слотов для ЛПУ – числовое поле;
- количество занятых слотов для ЕПГУ – числовое поле;

– количество занятых слотов для ЛПУ – числовое поле.

Объекты в оперативной памяти (временное хранение при работе программы):

– данные преобразуются в структуры Python (namedtuple или классы);
– группируются по категориям ЛПУ (ЦРБ, поликлиники, диспансеры).

Взаимосвязи и процессы.

Чтение данных:

– программа загружает Excel-файл, проверяет его структуру и извлекает данные;
– обрабатываются только разрешенные должности (фильтрация ненужных записей).

Агрегация и расчеты:

– общее количество слотов (total_slots = total_epgu + total_lpu);
– процент слотов для ЕПГУ (epgu_percent = (total_epgu / total_slots) * 100);
– уровень занятости (busy_rate = (busy_epgu / total_epgu) * 100);
– данные сортируются по группам ЛПУ для визуализации.

Передача в веб-интерфейс:

– результаты агрегации передаются в шаблоны Jinja2 в формате JSON;
– веб-страница отображает гистограммы и таблицы через Chart.js.

Преобразование данных:

данные из файла Excel преобразуются в объекты Python с помощью класса ReportRow, который представляет собой именованный кортеж. Каждая строка файла Excel соответствует объекту ReportRow, который затем используется для дальнейшей обработки и визуализации.

Логическая модель позволяет четко определить, как данные будут храниться и обрабатываться в системе, что упрощает разработку и тестирование.

2.3 Информационное обеспечение в АИС

Информационное обеспечение системы включает описание источников данных, их форматов и способов обработки.

Источники данных: основным источником данных являются файлы Excel, выгружаемые из системы РМИС (региональная медицинская информационная система). Эти файлы содержат информацию о расписании специалистов и количестве доступных слотов для записи.

Форматы данных: файлы Excel имеют строгую структуру, включающую следующие столбцы:

- название ЛПУ;
- должность специалиста;
- количество слотов для ЕПГУ;
- количество слотов для ЛПУ;
- количество занятых слотов для ЕПГУ;
- количество занятых слотов для ЛПУ.

Для работы с данными реализованы ETL-процессы (Extract, Transform, Load):

- извлечение: чтение Excel-файлов с помощью библиотеки xlrd, валидация структуры;
- преобразование: фильтрация записей, расчет агрегированных показателей (общее количество слотов, процент занятости);
- загрузка: передача данных в веб-интерфейс в формате JSON для визуализации.

Обработка ошибок включает проверку наличия файлов и корректности данных, а результаты логируются для отслеживания проблем.

Обработка данных: для чтения данных из файлов Excel используется библиотека xlrd. Данные из файлов загружаются в оперативную память, где обрабатываются и преобразуются в структуры данных Python для дальнейшей

визуализации.

Информационное обеспечение системы обеспечивает четкое понимание того, как данные поступают в систему, как они структурированы и как обрабатываются.

2.4 Требования к аппаратно-программному обеспечению АИС

Для работы системы необходимо следующее аппаратное и программное обеспечение.

Аппаратное обеспечение:

- компьютер с процессором не ниже Intel Core i3 или аналогичным,
- оперативная память не менее 4 ГБ,
- свободное место на диске не менее 500 МБ для хранения файлов и временных данных.

Программное обеспечение:

- операционная система: Windows 10 или выше, Linux, macOS,
- установленный Python версии 3.10.11,
- веб-браузер (например, Google Chrome, Mozilla Firefox) для отображения веб-интерфейса.

Эти требования обеспечивают корректную работу системы на компьютерах пользователей [21, 22].

Вывод по главе

Проработка всех пяти пунктов – от выбора технологии логического моделирования до определения требований к аппаратно-программному обеспечению – позволила создать комплексную и структурированную документацию, которая охватывает ключевые аспекты проектирования и разработки системы. Использование UML для создания диаграмм классов и последовательностей обеспечило наглядное представление архитектуры

системы, что значительно упростило процесс разработки и взаимодействие между компонентами. Логическая модель, основанная на структуре данных из файлов Excel, стала основой для обработки и визуализации данных, а также обеспечила гибкость и удобство работы с информацией.

Информационное обеспечение системы, включая описание источников данных, их форматов и процессов обработки, позволило минимизировать ошибки и упростить интеграцию с существующей инфраструктурой. Решение об отказе от использования базы данных в пользу обработки данных в оперативной памяти было обосновано временным характером данных и небольшим объемом информации. Однако, проект предусматривает возможность масштабирования с использованием базы данных, такой как SQLite или PostgreSQL, в случае увеличения объема данных или необходимости долгосрочного хранения.

Определение требований к аппаратно-программному обеспечению обеспечило стабильную работу системы на компьютерах пользователей. Минимальные требования к оборудованию и программному обеспечению позволяют развернуть систему на большинстве современных компьютеров, что делает ее доступной для широкого круга пользователей.

В целом, проработка всех пунктов позволила создать систему, которая эффективно решает поставленные задачи: автоматизирует процесс обработки и визуализации данных, экономит время пользователей и минимизирует ошибки, связанные с ручной обработкой информации. Документация, созданная в рамках проекта, обеспечивает прозрачность и понимание системы для всех участников, а также закладывает основу для возможного масштабирования и развития системы в будущем.

Глава 3 Физическое проектирование АИС

3.1 Выбор архитектуры АИС

В рамках ВКР разработан проект в котором была разработана архитектура веб-приложения для анализа и визуализации данных о доступности электронных записей в медицинских учреждениях [6, 23]. Система представляет собой комплексное решение, состоящее из следующих ключевых компонентов:

Веб-интерфейс (Dashboard):

- обеспечивает интерактивное отображение статистики через динамически формируемые HTML-страницы;
- использует современные технологии визуализации данных.

Бэкенд-обработчик (TimetableReportHandler):

- выполняет чтение и анализ данных из Excel-файлов;
- осуществляет сложные расчеты и агрегацию показателей.

Серверная часть (CherryPy):

- обеспечивает надежную обработку HTTP-запросов;
- координирует взаимодействие между всеми компонентами системы.

Архитектура системы [12, 16] демонстрирует четкое разделение ответственности между компонентами и оптимизированный поток данных от источников информации до конечного пользователя. Особое внимание уделено:

- простоте интеграции с существующей инфраструктурой;
- эффективности обработки временных данных;
- удобству визуализации сложных статистических показателей;
- сортировка данных по разделам.

Разработанное решение позволяет медицинским учреждениям оперативно анализировать загруженность системы электронной записи и

принимать обоснованные управленческие решения. Архитектурная схема приложения показана на рисунке 11.

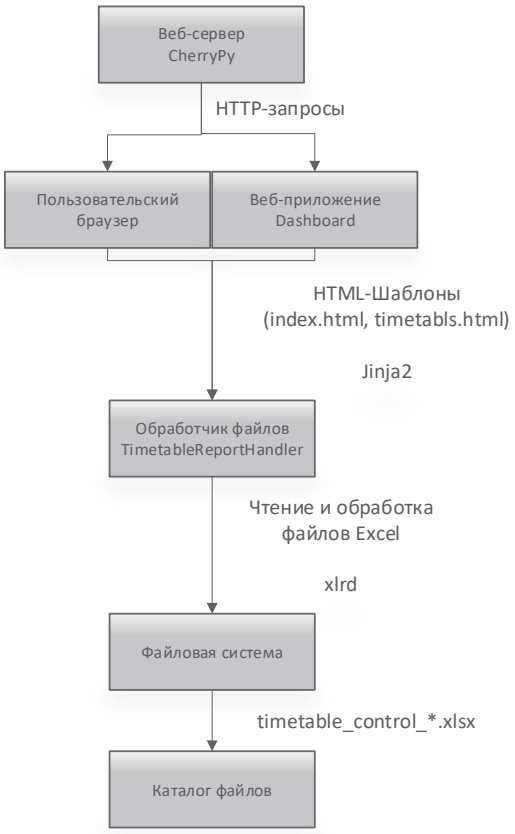

Рисунок 11 – Архитектурная схема приложения

Схема пользовательского взаимодействия детально отображает полный цикл работы конечного пользователя с аналитической системой. На схеме представлены:

- процесс загрузки исходных данных (excel-файлов с расписанием врачей);
- механизмы взаимодействия с веб-интерфейсом (кнопки управления, формы ввода);
- последовательность операций от отправки запроса до получения результатов;
- элементы обратной связи и статуса выполнения операций;

– адаптивные компоненты интерфейса для различных типов устройств.

Схема пользовательского взаимодействия показана на рисунке 12.

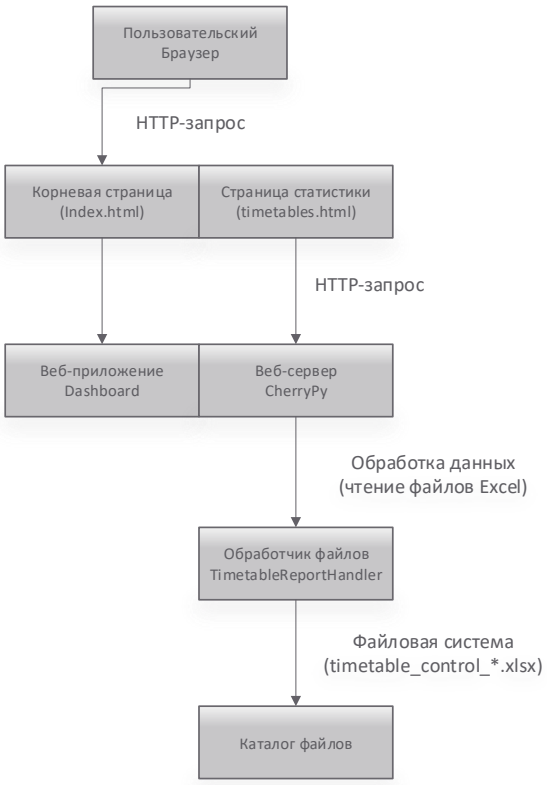

Рисунок 12 – Пользовательское взаимодействие

Схема обработки данных детально отображает архитектуру многоэтапного конвейера преобразования информации. На первом этапе система выполняет чтение и первичную проверку Excel-файлов, включая контроль целостности данных и соответствия требуемому формату. Затем данные проходят нормализацию – приведение к единому стандартизированному виду, что обеспечивает их последующую корректную обработку. Особое внимание уделено алгоритмам агрегации показателей по медицинским учреждениям, которые позволяют консолидировать разрозненные данные в целостную аналитическую картину. Ключевой составляющей является расчет метрик, в частности определение процента

доступных слотов ЕПГУ, что дает количественную оценку эффективности работы системы электронной записи. Завершающие этапы включают формирование структурированных наборов для визуализации и комплексную обработку возможных ошибок, гарантируя надежность и достоверность результатов. Схема обработки данных показана на рисунке 13.

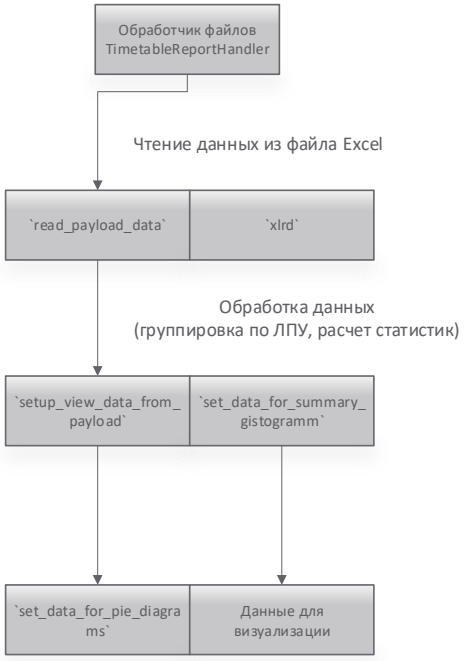

Рисунок 13 – Обработка данных

Схема визуализации данных преобразует обработанные данные в удобные для анализа формы. Основу составляют различные типы графиков: столбчатые диаграммы наглядно отображают распределение слотов, а круговые – пропорции разных типов записи. Интерактивные таблицы с функционалом сортировки и фильтрации позволяют детально изучать информацию. Цветовое кодирование и адаптивные форматы отображения, учитывающие специфику различных категорий ЛПУ, значительно упрощают восприятие данных. Система дополнена динамическими элементами управления для выбора периода и учреждений, а также контекстными

подсказками, помогающими пользователям ориентироваться в аналитике. Схема визуализации данных показана на рисунке 14.

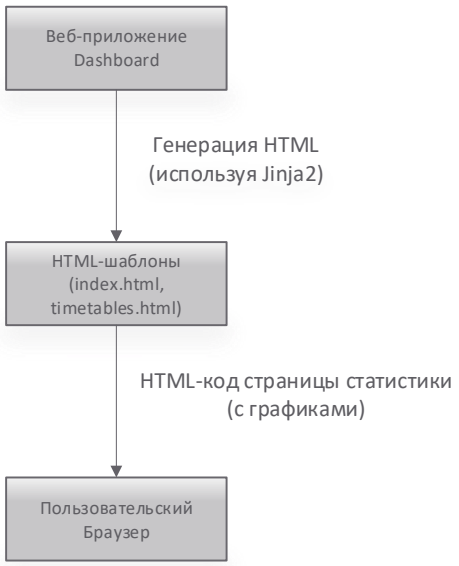

Рисунок 14 – Визуализация данных

Реализованный подход к обработке и визуализации данных обеспечивает комплексное решение для мониторинга и анализа доступности электронной записи в медицинских учреждениях. Интеграция всех компонентов позволяет не только автоматизировать рутинные процессы, но и предоставляет руководству удобный инструмент для принятия обоснованных управленческих решений.

3.2 Выбор технологии разработки АИС

При проектировании системы [11], представленной в приложенных файлах, были выбраны следующие решения, основываясь на их простоте, эффективности и подходящей функциональности для данного случая [25]:

Выбор веб-фреймворка CherryPy:

– простота: CherryPy – легкий и простой в использовании фреймворк,

который отлично подходит для небольших веб-приложений;

– гибкость: CherryPy предоставляет гибкость в настройке и расширении функциональности веб-приложения;
– эффективность: CherryPy относительно легковесен и эффективен в использовании ресурсов.

Использование шаблонизатора Jinja2:

– отделение логики от представления: Jinja2 позволяет отделить бизнес-логику от HTML-представления приложения, делая код более читаемым и удобным в разработке;
– гибкость: Jinja2 предоставляет широкие возможности для динамической генерации HTML с помощью шаблонов.

Использование библиотеки xlrd для работы с файлами Excel:

– простота использования: xlrd – простая и надежная библиотека для чтения файлов Excel в Python;
– доступность: xlrd – широко используемая и хорошо документированная библиотека.

Структура приложения:

– разделение на модули: приложение разделено на два модуля: file_handlers и dashboard. Такая структура позволяет упростить разработку и поддерживать код в порядке;
– классы: использование классов TimetableReportHandler и Dashboard помогает структурировать код и упрощает его рефакторинг.

Выбор подхода к обработке данных [17]:

– чтение данных из файлов Excel: выбор файлов Excel в качестве источника данных является простым и эффективным решением для данного случая;
– обработка данных в памяти: приложение читает данные из файлов Excel и обрабатывает их в памяти. Этот подход эффективен для небольших объемов данных.

Отсутствие сложной логики:

– минимальный функционал: приложение предоставляет простой и минимальный набор функциональности – чтение данных из файлов Excel и визуализация статистики;
– легкость сопровождения: отсутствие сложной логики упрощает сопровождение и модернизацию приложения.

Использование статического HTML:

– простота: использование статических HTML-шаблонов с динамическими вставками с помощью Jinja2 упрощает разработку и обслуживание приложения.

Основные преимущества данного решения:

– простота: простая архитектура, легкий в использовании код, минимальный набор функций;
– гибкость: возможность расширения функциональности с помощью CherryPy и Jinja2 [1];
– эффективность: приложение относительно легковесно и эффективно в использовании ресурсов.

Важно отметить:

– данная архитектура подходит для небольших веб-приложений с простым набором функций и небольшими объемами данных;
– для более сложных приложений может потребоваться более сложная архитектура с использованием более мощных веб-фреймворков, баз данных, API и др.

3.3 Разработка физической модели данных АИС

Физическая модель [9] описывает, как данные будут храниться и обрабатываться на физическом уровне. В данном проекте данные хранятся в оперативной памяти и файлах Excel, поэтому физическая модель [10]

описывает структуру этих данных.

Наполнение:

- хранение данных: данные хранятся в оперативной памяти в виде объектов Python (например, списки и словари), а также в файлах Excel;
- обработка данных: Описание того, как данные из файлов Excel загружаются в память, обрабатываются и передаются в веб-интерфейс для визуализации;
- производительность: учитывая, что данные хранятся в оперативной памяти, важно учитывать объем данных и производительность системы при работе с большими файлами Excel.

3.4 Разработка программного обеспечения АИС

В рамках проекта разработана программа [12, 16, 17] для визуализации отчета. Программа сделана таким образом, что из системы РМИС выгружается отчет по ведению расписания МО за необходимый период. Отчет помещается в определенную папку, после чего запускается сама программа и пользователь в удобном для него браузере вводит указанный в программе IP-адресс. После открывается страница, выбирается данный отчет и осуществляется переход на новую страницу с визуализацией данных в загруженном отчете, что экономит время на ручной сбор информации и визуализации ее. Данная программа разработана для руководства Министерства здравоохранения Вологодской области для оптимизации затрачиваемого времени на обработку данных используемых во время проведения ВКС с МО по показателям региона.

В работе приложения основными файлами являются:

- main.py (Приложение А) – реализует веб-приложение, которое отображает статистику по отчетам о расписании;
- file_handlers.py (Приложение Б) – обработчик файлов отчетов о расписании;

– timetables.html (Приложение В) – HTML-шаблон.

Описание работы файла main.py:

импорты:

– from dataclasses import dataclass: импортирует декоратор dataclass, который упрощает создание классов с данными;
– import os: импортирует модуль os, который предоставляет функции для работы с файловой системой;
– import cherrypy: импортирует веб-фреймворк CherryPy для создания веб-приложения;
– from pathlib import Path: импортирует класс Path, который предоставляет более удобный способ работы с путями к файлам;
– from os import path: импортируется модуль path из os, который предоставляет функции для работы с путями к файлам;
– from jinja2 import Environment, FileSystemLoader: импортируются классы Environment и FileSystemLoader из библиотеки Jinja2 для обработки шаблонов;
– from file_handlers import TimetableReportHandler: импортируется класс TimetableReportHandler из модуля file_handlers, который, предположительно, отвечает за чтение данных из файлов отчетов.

Класс LPUDataset:

– использует декоратор @dataclass для удобного создания класса, который представляет собой набор данных статистики по лечебно-профилактическому учреждению (ЛПУ);
– имеет поля для хранения названия ЛПУ (lpu_name), общего количества бирок (total_tickets_by_lpu), количества бирок для ЕПГУ (epgu), норматива бирок для ЕПГУ (epgu_norma), количества занятых бирок для ЕПГУ (epgu_busy) и процента бирок для ЕПГУ от общего количества (percent).

Класс Dashboard:

– класс Dashboard реализует веб-приложение для отображения статистики.

Конструктор:

– устанавливает базовые пути к шаблонам (BASE_TEMPLATE_PATH) и файлам отчетов (BASE_FILEREPORTS_PATH);
– создает объект Environment для обработки шаблонов Jinja2.

Метод index:

– обрабатывает запрос на корневую страницу приложения;
– возвращает HTML-шаблон index.html.

Метод timetables:

– обрабатывает запрос на страницу со статистикой по биркам расписания;
– перебирает файлы в каталоге отчетов, используя os.listdir;
– извлекает период отчета из названия файла;
– создает объект TimetableReportHandler для обработки файла отчета;
– формирует список diagrams с данными по группам ЛПУ;
– обновляет контекст (context) для передачи данных в шаблон;
– возвращает HTML-шаблон timetables.html.

Основной код [13, 14, 24]:

– если код запущен непосредственно, создается и запускается веб-приложение CherryPy с классом Dashboard().

Описание работы timetables:

метод timetables осуществляет следующие действия:

– перебирает файлы в каталоге BASE_FILEREPORTS_PATH, используя os.listdir;
– проверяет, содержит ли имя файла подстроку timetable_control_, чтобы определить, является ли этот файл отчетом;
– извлекает из имени файла период отчета и сохраняет его в контексте

context;

- создает объект TimetableReportHandler, передавая путь к файлу отчета;
- формирует список diagrams с данными по группам ЛПУ;
- сортирует данные по группам ЛПУ по проценту бирок для ЕПГУ;
- обновляет контекст context данными отчета, список специальностей, общей статистикой, данными для круговых диаграмм и данными для гистограмм (разделенных по группам);
- возвращает HTML-шаблон timetables.html с данными из контекста.

Описание работы файла file_handlers.py:

импорты:

- from __future__ import annotations: позволяет использовать аннотации типов в соответствии с новым стандартом Python 3.10, улучшая читаемость и контроль типов в коде;
- from collections import namedtuple: импортирует класс namedtuple, который позволяет создавать именные кортежи, удобные для хранения и доступа к данным;
- import os: импортирует модуль os, который предоставляет функции для работы с файловой системой;
- import xlrd: импортирует модуль xlrd, который предоставляет функции для чтения файлов Excel.

Определение типа данных ReportRow:

ReportRow = namedtuple('ReportRow', ['name', 'post', 'total_for_epgu', 'total_for_lpu', 'busy_for_epgu', 'busy_for_lpu',]): Создает именной кортеж ReportRow, который будет использоваться для представления данных из отчета в виде структуры с именованными полями:

- name: название ЛПУ;
- post: должность;
- total_for_epgu: общее количество слотов для ЕПГУ;

– total_for_lpu: общее количество слотов для ЛПУ;
– busy_for_epgu: количество занятых слотов для ЕПГУ;
– busy_for_lpu: количество занятых слотов для ЛПУ.

Класс TimetableReportHandler:

– LPU_GROUPS: список групп ЛПУ (ЦРБ, поликлиники, диспансеры/спец. МО) с указанием названий ЛПУ, входящих в каждую группу;
– START_READ_ROW_NUMBER: определяет номер строки в файле Excel, с которой начинается чтение данных (с 3-й строки);
– ALLOWED_POSTS: список разрешенных должностей, данные по которым будут обрабатываться;
– __init__: конструктор класса, который принимает путь к файлу отчета в качестве аргумента report_path;

Функции:

– проверяет существование файла по указанному пути и выбрасывает исключение RuntimeError, если файл не найден;
– создает список payload для хранения данных из отчета;
– вызывает метод read_payload_data, чтобы прочитать данные из файла отчета и заполнить список payload;
– вызывает методы setup_view_data_from_payload, set_data_for_summary_gistogramm и set_data_for_pie_diagrams, чтобы подготовить данные для визуализации.

Методы класса TimetableReportHandler:

setup_view_data_from_payload:

– сортирует данные из отчета по ЛПУ, считает общее количество бирок и заполняет словарь data_by_lpu, который будет использоваться для отображения данных на странице;
– также подсчитывает общее количество бирок, количество бирок для ЕПГУ и количество занятых бирок для ЕПГУ.

set_data_for_summary_gistogramm:

– собирает данные для основной гистограммы, подсчитывая общее количество бирок, количество бирок для ЕПГУ и количество занятых бирок для ЕПГУ по каждому ЛПУ.

set_data_for_pie_diagrams:

– собирает данные для круговых диаграмм, рассчитывая проценты бирок для ЕПГУ и занятых бирок для ЕПГУ от общего количества бирок по каждому ЛПУ.

read_payload_data:

– читает данные из файла отчета Excel и заполняет список payload объектами ReportRow;
– использует xlrd для чтения данных из файла Excel.

В целом, класс TimetableReportHandler представляет собой обработчик файлов отчетов о расписании. Он читает данные из файла Excel, обрабатывает их и предоставляет структурированные данные для визуализации на веб-странице.

Описание работы файла timetables.html: этот файл представляет собой HTML-шаблон для веб-приложения, которое визуализирует статистику по биркам расписания. Он использует фреймворк Jinja2 для динамической генерации HTML-кода.

Шапка HTML-документа:

– определяется DOCTYPE, язык HTML (lang="ru") и кодировка utf-8;
– устанавливается заголовок страницы Графические отчеты;
– подключается CSS для стилизации страницы, а также библиотеки Chart.js и chartjs-plugin-datalabels для построения графиков;
– подключается библиотека semantic-ui для стилизации элементов страницы.

Основной контент:

– заголовок: отображается заголовок Статистика по биркам: {{ period

}}, где {{ period }} – динамическая переменная, которую будет заменять Jinja2 в зависимости от данных отчета;

- информация о специальностях: отображается список специальностей (взятый из allowed_posts), которые учитываются в отчете;
- блок общей статистики: отображается общая статистика по биркам: общее количество, количество бирок для ЕПГУ и количество бирок для ЛПУ;
- гистограммы: для каждой группы ЛПУ (ЦРБ, поликлиники, диспансеры/спец. МО) отображается гистограмма;
- для каждой гистограммы генерируется canvas и соответствующий скрипт Chart.js с данными и настройками графика;
- в скрипте используются переменные из контекста Jinja2, например, {{ diagram.lpu_datasets }} для получения данных о ЛПУ из контекста;
- гистограмма отображает количество слотов для ЕПГУ, общее количество слотов и количество занятых слотов для ЕПГУ по каждому ЛПУ.

Как работает этот файл:

- HTML-шаблон используется веб-приложением для отображения статистики по отчетам о расписании;
- веб-приложение получает данные из отчета Excel с помощью TimetableReportHandler и затем передает эти данные в контекст Jinja2;
- Jinja2 заменяет динамические переменные в HTML-шаблоне данными из контекста, генерируя HTML-код страницы со статистикой;
- браузер пользователя отображает сгенерированный HTML-код с графиками.

Для обеспечения стабильной работы проекта было создано изолированное виртуальное окружение с помощью команды python -m venv env (рисунок 15). Использование виртуального окружения позволяет избежать конфликтов версий пакетов и гарантирует воспроизводимость среды разработки на разных компьютерах. Особое внимание уделялось выбору версии Python 3.10.11, что обусловлено необходимостью совместимости с пакетом python-cql, который имеет ограничения в работе с более новыми версиями интерпретатора.

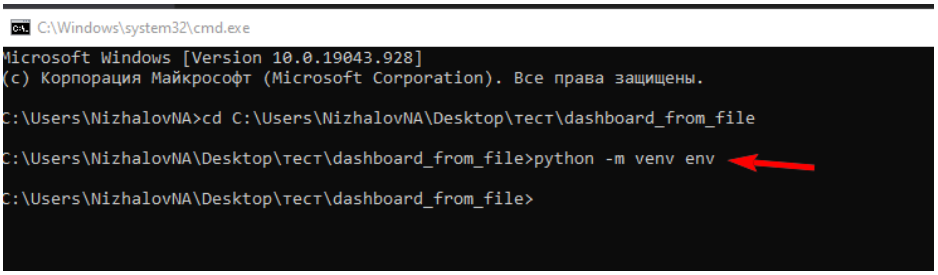

Рисунок 15 – Создание виртуального окружения

После создания виртуального окружения выполнена установка всех необходимых для работы библиотек. Для удобства управления зависимостями использовался файл requirements.txt, содержащий точные версии всех требуемых пакетов. Такой подход обеспечивает простоту развертывания системы на новых рабочих местах и гарантирует идентичность окружения. Среди ключевых установленных библиотек следует отметить CherryPy для веб-сервера, Jinja2 для шаблонов и xlrd для работы с Excel-файлами.

Среди ключевых установленных библиотек следует отметить:

– CherryPy 18.6.0 – легковесный веб-сервер для развертывания приложения;
– Jinja2 3.0.3 – система шаблонов для динамической генерации HTML;
– xlrd 2.0.1 – надежная библиотека для чтения Excel-файлов;

– matplotlib 3.5.1 – инструмент для построения графиков и диаграмм;

– numpy 1.21.2 – обеспечивает поддержку математических операций.

Установка библиотек показана на рисунке 16.

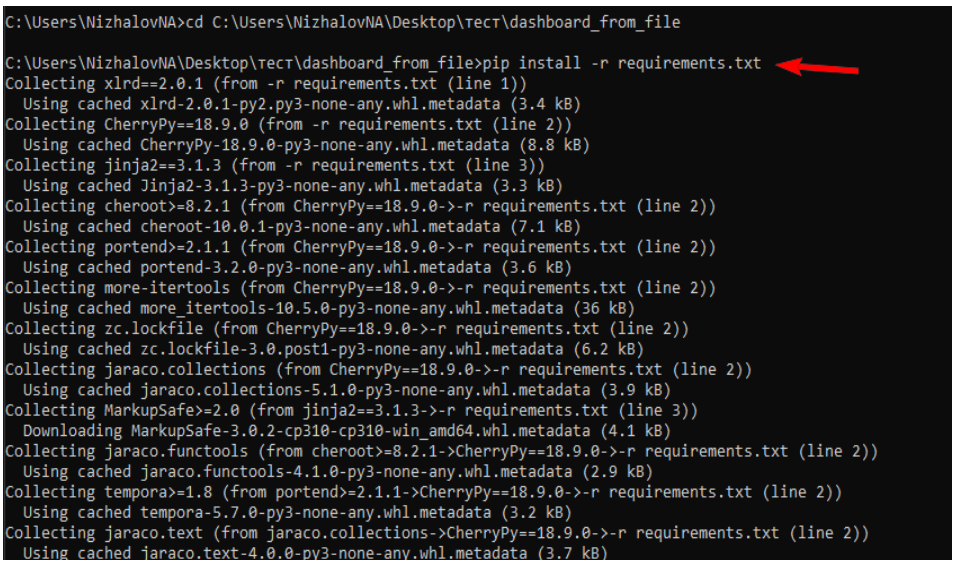

Рисунок 16 – Установка библиотек

На этапе подготовки к распространению приложения был установлен специализированный плагин auto-py-to-exe. Этот инструмент был выбран благодаря своей простоте использования и широким возможностям по настройке процесса конвертации. Ключевыми критериями выбора стали: поддержка всех необходимых функций упаковки, простота конфигурации, а также возможность создания автономного исполняемого файла, не требующего установки дополнительных компонентов на компьютере пользователя. После установки плагин был запущен и настроен для работы с текущим проектом. После установки проведена верификация работоспособности плагина через выполнение тестовой сборки простого приложения. Установка плагина auto-py-to-exe показана на рисунке 17.

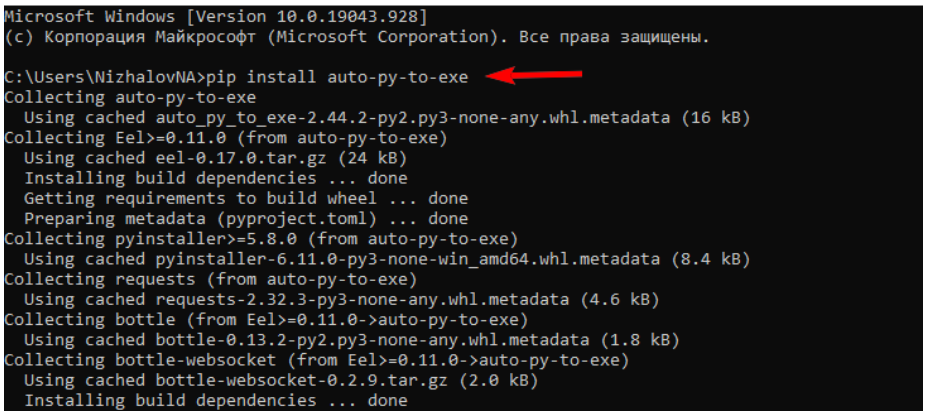

Рисунок 17 – Установка плагина auto-py-to-exe

После запуска плагина в браузере открывается вкладка, в которой выбираются необходимые настройки. Процесс конвертации Python-скрипта в исполняемый файл осуществлялся с использованием инструмента auto-py-to-exe, который обеспечил преобразование основного скрипта main.py в автономный исполняемый файл main.exe. На этапе настройки были определены ключевые параметры: выбран режим "Onefile" для создания единого компактного файла, указаны пути к основным ресурсам проекта, заданы параметры оптимизации и сжатия, а также настроена иконка приложения. Особое внимание уделялось корректной обработке зависимостей - автоматическому сканированию и включению всех необходимых библиотек с последующей ручной проверкой на предмет избыточных модулей. В процессе сборки происходила комплексная обработка импортов, оптимизация байт-кода и генерация временных файлов в изолированной директории. После завершения конвертации выполнялась тщательная верификация полученного EXE-файла, включающая проверку целостности, тестирование на чистой системе и валидацию всех функциональных сценариев работы приложения. Результатом стал компактный исполняемый файл размером около 15 МБ, содержащий интерпретатор Python, необходимые библиотеки, HTML-шаблоны и конфигурационные файлы, что позволяет запускать приложение на любом компьютере с Windows без дополнительных требований к установке

программного обеспечения. Заполнение данных по конвертации документа показано на рисунке 18.

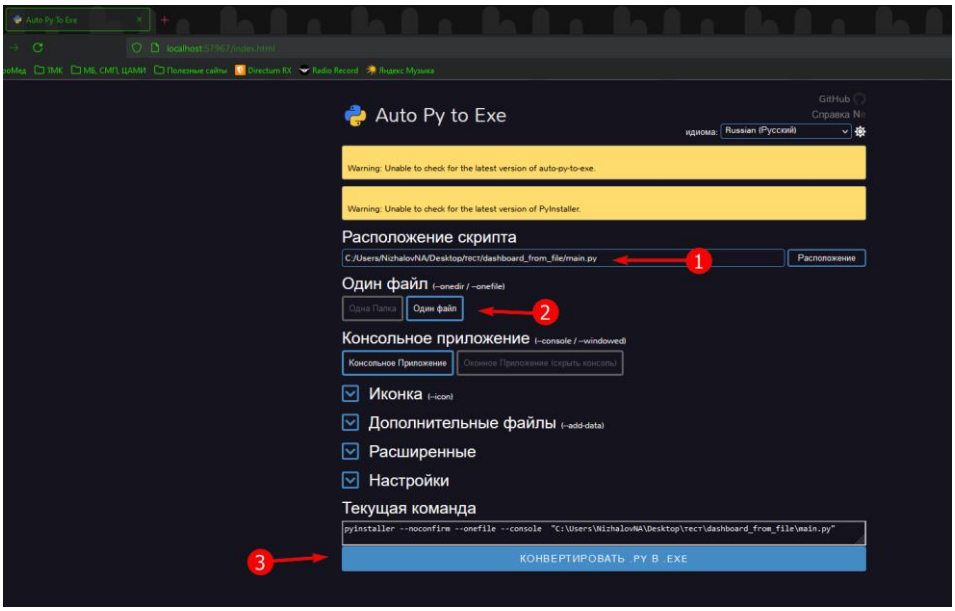

Рисунок 18 – Конвертация файла из py в exe

Информация о преобразовании файла из py в exe показана в Приложении Г.

После получения файла main.exe в папку с данным файлом создаются 2 папки: src и templates. В папке src находится xls файл, данные которого будут визуализироваться, а в папке templates находятся шаблоны html файлов.

В проекте используется Python 3.10.11 (рекомендуемое 3.8-3.12), т.к. необходима работа с пакетом python-cql, а в python 3.13 есть проблема с работой данного пакета. Так же были установлены необходимые библиотеки и создана виртуальная среда venv для использования текущей версии пакетов python. В результате запуска auto-py-to-exe был скомпилирован файл main.py, выстроены необходимые взаимодействия и результатом сформирован файл main.exe, который позволяет работать данной программе на компьютере пользователя без установки дополнительных компонентов, а именно самого python и необходимых для работы дополнительных библиотек.

3.5 Описание функциональности АИС

Программа разработана с учетом максимального удобства для конечного пользователя. Начальный этап работы предполагает выгрузку отчета «Контроль специалистов и расписаний» из системы РМИС за требуемый период (рисунок 19). Для примера взят временной диапазон с 23.10.2024 по 30.10.2024, что соответствует стандартной практике еженедельного мониторинга показателей. Выгружаемый отчет содержит актуальные данные о доступности записей к специалистам медицинских учреждений.

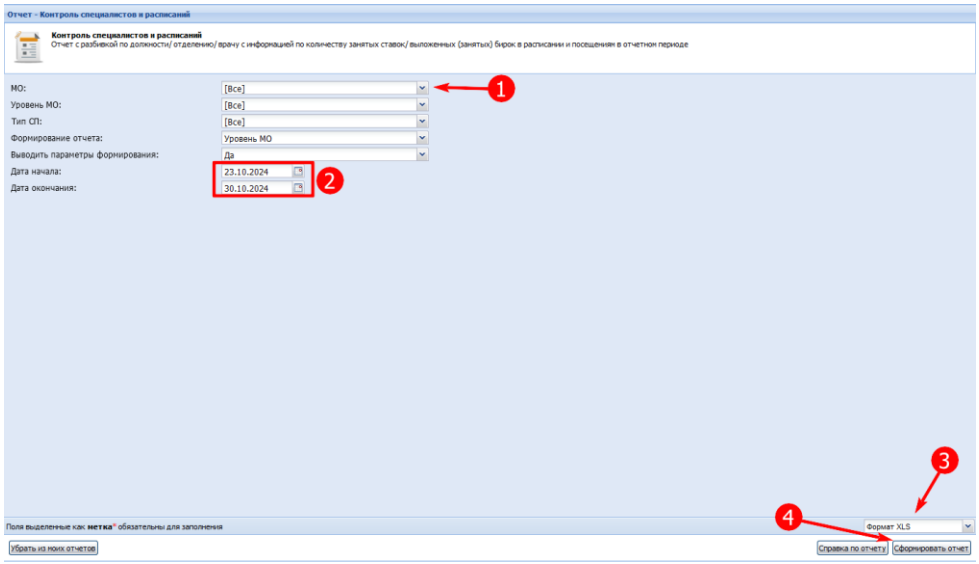

Рисунок 19 – Формирование отчета

После получения отчета пользователю необходимо разместить файл в специально предназначенной для этого папке src (рисунок 19). Важным требованием является приведение имени файла к стандартному формату timetable_control_ДД_ММ_ДД_ММ.xls, где первые две группы цифр обозначают дату начала периода, а последующие - дату окончания. Такая стандартизация именования позволяет программе автоматически

распознавать и корректно обрабатывать временные периоды, указанные в отчете.

Для начала работы с данными пользователь запускает исполняемый файл программы (рисунок 20). Приложение автоматически разворачивает локальный веб-сервер и выводит в консоли статический IP-адрес (127.0.0.1:8080) для доступа к интерфейсу. Использование локального адреса гарантирует безопасность работы с конфиденциальными медицинскими данными, исключая их передачу по сети.

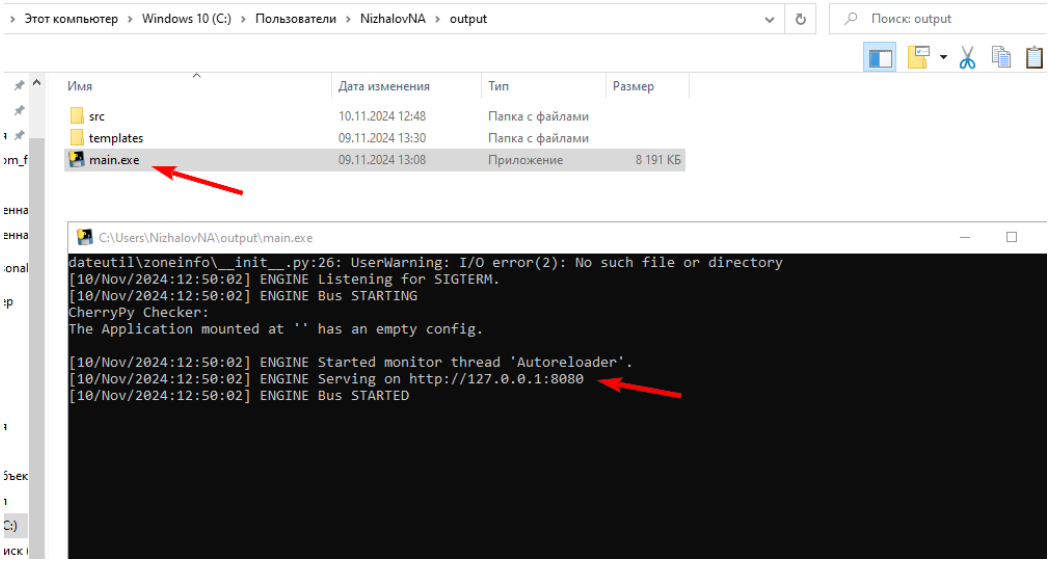

Рисунок 20 – Запуск приложения и отображение IP-адреса

Данный IP-адрес является статическим (127.0.0.1:8080), что позволяет пользователю сохранить его в браузере. После открытия страницы, пользователь нажимает на ссылку «Дашборд с расписаниями».

Система предоставляет многоуровневую аналитику показателей доступности записей. На первом уровне отображается сводная статистика по всему региону, включающая общие цифры по количеству доступных и занятых слотов. Далее данные детализируются по категориям медицинских учреждений: центральные районные больницы (ЦРБ), поликлиники и специализированные диспансеры. Отображение общей статистики и в разрезе

ЦРБ показано на рисунке 21.

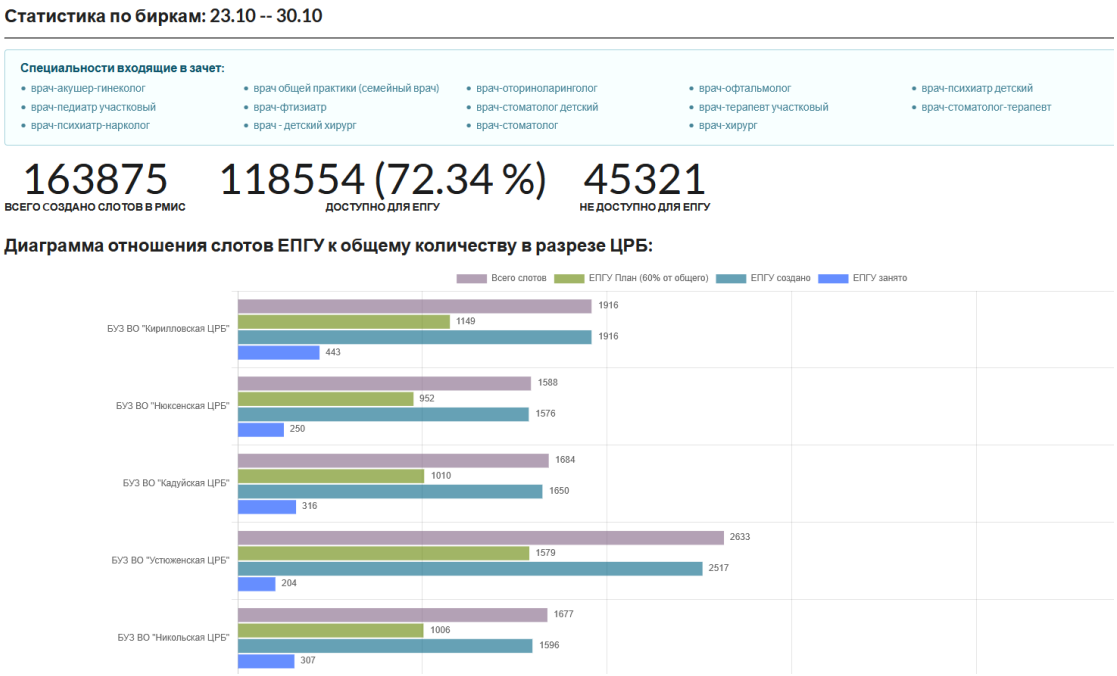

Рисунок 21 – Общая статистика региона

Пользователь может гибко настраивать отображение информации, скрывая или акцентируя определенные данные. Это особенно полезно при работе с большими массивами информации, когда необходимо сосредоточиться на конкретных показателях. Все изменения интерфейса сохраняются в течение сеанса работы, но не влияют на исходные данные.

По окончании анализа пользователь закрывает программу, что автоматически останавливает локальный сервер и освобождает используемые ресурсы. В консоли приложения фиксируется вся информация о работе системы, включая время обращения к данным и выполненные операции. Эти логи могут быть использованы для последующего анализа использования системы.

3.6 Оценка и обоснование экономической эффективности разработки АИС

В данной ситуации целесообразно использовать метод рентабельности инвестиций (ROI), так как проект не предполагает непосредственную продажу продукта или услуги, а направлен на повышение эффективности существующей системы. ROI покажет, насколько эффективно использованы инвестированные средства в проект. Формула расчета показана ниже:

$$ROI = \frac{Экономическая\ выгода\text{-}Затраты\ на\ проект}{Затраты\ на\ проект} \quad (1)$$

Исходные данные.

Текущие затраты на ручную обработку [2, 5]:

- время на обработку данных: 8 часов/месяц (2часа/неделю);
- зарплата специалиста: 50 000 руб./месяц;
- стоимость 1 часа работы:

$$1\ час\ работы = \frac{50\ 000 руб.}{20\ дней * 8\ часов} = 312{,}5\ руб./час; \quad (2)$$

- годовые затраты на ручную обработку:
-

$$годовые\ затраты = 8\ часов/месяц * 12 * 312{,}5\ руб = 30000\ руб./год; \quad (3)$$

Затраты на внедрение АИС:

- внутренняя разработка: 0 руб. (использованы собственные ресурсы);
- альтернатива (закупка): 10 000 000 руб.

Экономия от внедрения:

– сокращение времени на 98%: экономия 29 400 руб./год (98% от 30 000 руб.).

Расчет экономической эффективности внутренней разработки:

– годовой экономический эффект: 29 400 руб./год (полная экономия, т.к. нет затрат на разработку);
– срок окупаемости: мгновенный;
– коэффициент эффективности(K):K = Экономия/затраты = 29 400 / 0 (стремиться к бесконечности).

Расчет экономической эффективности закупки готового решения:

– годовой экономический эффект: 29 400 руб./год – 10 000 000 = -9 970 600 руб./год;
– срок окупаемости: 10 000 000/29 400 ≈ 340 лет;
– ROI: (29 400 – 10 000 000)/10 000 000 = -0,997

$$ROI = \frac{29\,400 - 10\,000\,000}{10\,000\,000} = -0{,}997. \quad (3)$$

Визуализация ROI (рисунок 22).

На представленном графике наглядно показана разница в рентабельности инвестиций между двумя вариантами реализации проекта:

– закупка у разработчиков: ROI = -0,997 (крайне невыгодный вариант, убыточный);
– внутренняя разработка: ROI = 2,94 (высокая рентабельность благодаря минимальным затратам).

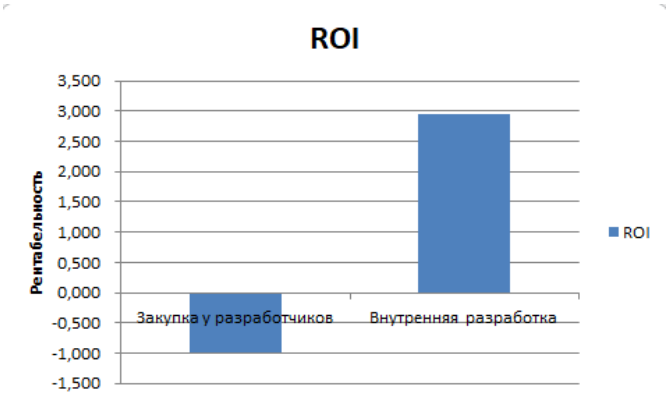

Рисунок 22. График разницы ROI по реализации проекта

Графическое сопоставление демонстрирует, что внутренняя разработка не только окупается, но и приносит значительную экономическую выгоду (294% от условных затрат), в то время как закупка готового решения ведет к финансовым потерям.

Вывод по главе

Разработанная АИС успешно решает поставленные задачи, обеспечивая автоматизацию обработки и визуализации данных, что значительно экономит время и ресурсы. Архитектура и технологии, выбранные для разработки, обеспечивают простоту, гибкость и эффективность системы. Оценка экономической эффективности подтверждает, что проект является выгодным и оправдывает затраты на его реализацию. Система готова к использованию и может быть легко масштабирована в случае увеличения объема данных или необходимости дополнительной функциональности.

Заключение

Разработанная автоматизированная информационная система для анализа данных о доступности электронных записей в медицинских учреждениях успешно решает поставленные задачи и демонстрирует высокую эффективность. В ходе работы создана комплексная система, позволяющая:

- автоматизировать процесс обработки данных из РМИС, исключив ручной труд и связанные с ним ошибки;
- обеспечить наглядную визуализацию ключевых показателей по записи пациентов через ЕПГУ;
- оптимизировать подготовку отчетов для видеоконференций с руководством здравоохранения региона.

Основные преимущества системы включают:

- простоту интеграции с существующей ИТ-инфраструктурой (работа с файлами Excel);
- интуитивно понятный интерфейс, не требующий специального обучения;
- быстроту обработки данных и формирования отчетов;
- гибкость архитектуры, позволяющую адаптировать систему под изменяющиеся требования.

Реализация проекта показала, что внутренняя разработка специализированного решения оказалась значительно эффективнее, чем закупка готового продукта. Система полностью соответствует потребностям организации и учитывает специфику работы с медицинскими данными.

Внедрение данной системы в БУ ВО «Медицинские цифровые технологии» позволило существенно повысить эффективность работы с данными о доступности электронных записей и качество принимаемых управленческих решений. Проект демонстрирует успешный пример цифровизации ключевых процессов в сфере здравоохранения.

Список используемой литературы

1. Бек К. Экстремальное программирование. – М.: Вильямс, 2002. – 224 с.
2. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. – М.: Символ-Плюс, 2015. – 304 с.
3. Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя. – М.: ДМК Пресс, 2010. – 496 с.
4. Beck, K. Extreme Programming Explained: Embrace Change. – Addison-Wesley, 2000. – 224 p.
5. Brooks, F. The Mythical Man-Month: Essays on Software Engineering. – AddisonWesley, 1995. – 322 p.
6. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – М.: Питер, 2017. – 368 с.
7. ГОСТ 34.201-89. Виды, комплектность и обозначение документов при создании автоматизированных систем. – М.: Издательство стандартов, 1989. – 12 с.
8. ГОСТ 34.602-89. Техническое задание на создание автоматизированной системы. – М.: Издательство стандартов, 1989. – 18 с.
9. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – М.: Вильямс, 2013. – 1328 с.
10. Крёнке Д. Теория и практика построения баз данных. – СПб.: Питер, 2003. – 800 с.
11. Липаев В.В. Проектирование программных средств. – М.: Финансы и статистика, 2008. – 320 с.
12. Martin, R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Prentice Hall, 2017. – 432 p.
13. Макконнелл С. Совершенный код. – М.: Русская редакция, 2010. – 896 с.

14. McConnell, S. Code Complete: A Practical Handbook of Software Construction. – Microsoft Press, 2004. – 960 p.

15. Маклаков С.В. Моделирование бизнес-процессов с BPwin 4.0. – М.: Диалог-МИФИ, 2002. – 224 с.

16. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – М.: Питер, 2018. – 352 с.

17. Орлов С.А. Технологии разработки программного обеспечения. – СПб.: Питер, 2012. – 464 с.

18. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – М.: Питер, 2013. – 896 с.

19. Соммервилл И. Инженерия программного обеспечения. – М.: Вильямс, 2011. – 752 с.

20. Sommerville, I. Software Engineering. – Pearson, 2016. – 792 p.

21. Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация. – СПб.: Питер, 2007. – 704 с.

22. Tanenbaum, A., Bos, H. Modern Operating Systems. – Pearson, 2014. – 1136 p.

23. Фаулер М. Архитектура корпоративных программных приложений. – М.: Вильямс, 2006. – 544 с.

24. Хант Э., Томас Д. Программист-прагматик. Путь от подмастерья к мастеру. – М.: Лори, 2004. – 288 с.

25. Шилдт Г. Java: руководство для начинающих. – М.: Вильямс, 2019. – 816 с.

Приложение А

Программный код файла main.py

```
from dataclasses import dataclass
import os
import cherrypy
from pathlib import Path
from os import path
from jinja2 import Environment, FileSystemLoader
from file_handlers import TimetableReportHandler


@dataclass
class LPUDataset:
    """Набор данных статистики по ЛПУ."""
    lpu_name: str
    total_tickets_by_lpu: int
    epgu: int
    epgu_norma: int
    epgu_busy: int
    percent: int


class Dashboard(object):
    """Класс приложения для отображения статистики."""

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.BASE_TEMPLATE_PATH: Path = Path(os.getcwd()) / 'templates'
        self.BASE_FILEREPORTS_PATH: Path = Path(os.getcwd()) / 'src'
        self.templenv =
Environment(loader=FileSystemLoader(str(self.BASE_TEMPLATE_PATH)))

    @cherrypy.expose
    def index(self):
        """Возвращает страницу со списком отчетов."""
        return self.templenv.get_template('index.html').render()

    @cherrypy.expose
    def timetables(self):
        """Возвращает страницу со статистикой по биркам расписания."""
        context = {}
        for item in os.listdir(str(self.BASE_FILEREPORTS_PATH)):
            if 'timetable_control_' in item:
                # Получаем период по названию отчета.
                fname_parts = item.replace('timetable_control_', '').split('.')[0].split('_')
                start_day, start_month, finish_day, finish_month = fname_parts
                context['period'] = f'{start_day.zfill(2)}.{start_month.zfill(2)} --
{finish_day.zfill(2)}.{finish_month.zfill(2)}'
```

```
                # Обрабатываем содержимое отчета.
                handler = TimetableReportHandler(str(self.BASE_FILEREPORTS_PATH / item ))

            # Формирование данных по группам.
            diagrams = []
            for group in handler.LPU_GROUPS:
                lpu_names_data = [x for x in handler.data_by_lpu if x in group['lpu_names']]

                group_name = group['name']
                calc_height = 35 * len(lpu_names_data)

                def sortSecond(val):
                    return val.percent

                lpu_datasets = []
                for lpu_name in lpu_names_data:
                    lpu_datasets.append(
                        LPUDataset(
                            lpu_name=lpu_name,
                            total_tickets_by_lpu=handler.total_tickets_diagram_data[lpu_name],
                            epgu=handler.epgu_ticket_diagram_data[lpu_name],
                            epgu_norma=int(handler.total_tickets_diagram_data[lpu_name] * 0.6),
                            epgu_busy=handler.epgu_busy_ticket_diagram_data[lpu_name],
                            percent=round(handler.epgu_ticket_diagram_data[lpu_name] * 100 / handler.total_tickets_diagram_data[lpu_name])
                        )
                    )
                lpu_datasets.sort(key=sortSecond, reverse=True)

                diagrams.append(
                    {
                        'group_name': group_name,
                        'calc_height': calc_height,
                        'lpu_datasets': lpu_datasets,
                    }
                )

            context.update(
                {
                    # Список специальностей.
                    'allowed_posts': TimetableReportHandler.ALLOWED_POSTS,

                    # Блок общей статистики.
                    'total': handler.amount_tickets,
                    'total_epgu': handler.amount_tickets_epgu,
                    'total_epgu_percent': round(handler.amount_tickets_epgu * 100 /
```

```
handler.amount_tickets, 2),
                    'total_lpu': handler.amount_tickets_lpu,

                    # Данные к круговым диаграммам.
                    'data_for_pie_diagrams': handler.pie_data,

                    # Данные к гистограммам, разделенные по группам.
                    'diagrams': diagrams,
                }
            )

        return self.templenv.get_template('timetables.html').render(context)


if __name__ == '__main__':
    cherrypy.quickstart(Dashboard())
```

Приложение Б

Программный код файла file_handlers.py

```
from __future__ import annotations
from collections import namedtuple
import os
import xlrd


ReportRow = namedtuple('ReportRow', ['name', 'post', 'total_for_epgu', 'total_for_lpu', 'busy_for_epgu', 'busy_for_lpu',])


class TimetableReportHandler:
    """Класс обработчик отчета по биркам."""

    LPU_GROUPS = [
        {
            'name': 'Диаграмма отношения слотов ЕПГУ к общему количеству в разрезе ЦРБ',
            'lpu_names': [
                'БУЗ ВО "Бабаевская ЦРБ"',
                'БУЗ ВО "Бабушкинская ЦРБ"',
                'БУЗ ВО "Белозерская ЦРБ"',
                'БУЗ ВО "Вашкинская ЦРБ"',
                'БУЗ ВО "Великоустюгская ЦРБ"',
                'БУЗ ВО "Верховажская ЦРБ"',
                'БУЗ ВО "Вожегодская ЦРБ"',
                'БУЗ ВО "Вытегорская ЦРБ"',
                'БУЗ ВО "Вологодская ЦРБ"',
                'БУЗ ВО "Грязовецкая ЦРБ"',
                'БУЗ ВО "Кадуйская ЦРБ"',
                'БУЗ ВО "Кирилловская ЦРБ"',
                'БУЗ ВО "Кич-Городецкая ЦРБ"имени В.И.Коржавина',
                'БУЗ ВО "Междуреченская ЦРБ"',
                'БУЗ ВО "Никольская ЦРБ"',
                'БУЗ ВО "Нюксенская ЦРБ"',
                'БУЗ ВО "Сокольская ЦРБ"',
                'БУЗ ВО "Сямженская ЦРБ"',
                'БУЗ ВО "Тарногская ЦРБ"',
                'БУЗ ВО "Тотемская ЦРБ"',
                'БУЗ ВО "Усть-Кубинская ЦРБ"',
                'БУЗ ВО "Устюженская ЦРБ"',
                'БУЗ ВО "ХАРОВСКАЯ ЦРБ"',
                'БУЗ ВО "Чагодощенская ЦРБ"',
                'БУЗ ВО "Шекснинская ЦРБ"',
            ]
        },
        {
            'name': 'Диаграмма отношения слотов ЕПГУ к общему количеству в разрезе
```

```
Поликлиник',
                'lpu_names': [
                    'БУЗ ВО "ВГСП"',
                    'БУЗ ВО "Вологодская городская поликлиника № 1"',
                    'БУЗ ВО "Вологодская городская поликлиника № 2"',
                    'БУЗ ВО "Вологодская городская поликлиника № 3"',
                    'БУЗ ВО "Вологодская городская поликлиника №4"',
                    'БУЗ ВО "Вологодская ГП №5"',
                    'БУЗ ВО "Череповецкая ГП № 1"',
                    'БУЗ ВО "Череповецкая городская поликлиника № 2"',
                    'БУЗ ВО "Череповецкая городская поликлиника № 7" им. П.Я. Дмитриева',
                    'БУЗ ВО "Череповецкая детская городская поликлиника № 1"',
                    'БУЗ ВО "Череповецкая ДГП № 3"',
                    'БУЗ ВО "Череповецкая детская стоматологическая поликлиника"',
                    'БУЗ ВО "Череповецкая стоматологическая поликлиника № 1"',
                    'БУЗ ВО "Череповецкая стоматологическая поликлиника № 2"',
                ]
            },
            {
                'name': 'Диаграмма отношения слотов ЕПГУ к общему количеству в разрезе
Диспансеров/Спец. МО',
                'lpu_names': [
                    'БУЗ ВО "ВОДБ №2"',
                    'БУЗ ВО "ВОДКБ"',
                    'БУЗ ВО "ВОКБ"',
                    'БУЗ ВО "Вологодская ОКБ № 2"',
                    'БУЗ ВО "Вологодская городская больница № 2"',
                    'БУЗ ВО "Вологодский ГРД"',
                    'БУЗ ВО "ВООБ"',
                    'БУЗ ВО "Вологодский областной онкологический диспансер"',
                    'БУЗ ВО "ВОПД"',
                    'БУЗ ВО Медико-санитарная часть "Северсталь"',
                    'БУЗ ВО "Череповецкая городская больница"',
                    'БУЗ ВО "Череповецкий городской родильный дом"',
                ]
            }
        ]

        START_READ_ROW_NUMBER = 3
        ALLOWED_POSTS = {
            'врач-хирург',
            'врач-офтальмолог',
            'врач-оториноларинголог',
            'врач-акушер-гинеколог',
            'врач-психиатр-нарколог',
            'врач-фтизиатр',
            'врач-стоматолог',
```

```
            'врач-педиатр участковый',
            'врач – детский хирург',
            'врач-стоматолог детский',
            'врач-психиатр детский',
            'врач-стоматолог-терапевт',
            'врач-терапевт участковый',
            'врач общей практики (семейный врач)',
        }

        def __init__(self, report_path: str) -> None:
            if not os.path.exists(report_path):
                raise RuntimeError("Файл отчета не существует по указанному пути %s" %
report_path)

            # Обрабатываем файл отчета
            self.payload: list[ReportRow] = []
            self.read_payload_data(report_path)

            self.setup_view_data_from_payload()

            # Данные для столбчатой диаграммы по биркам в разрезе ЕПГУ/МО.
            self.set_data_for_summary_gistogramm()
            # Данные для круговой диаграммы по всем организациям.
            self.set_data_for_pie_diagrams()

        def setup_view_data_from_payload(self):
            """Собирает набор данных необходимых для отображения на странице."""
            print('Сортируем по ЛПУ.')
            self.data_by_lpu = {}
            self.amount_tickets = 0
            self.amount_tickets_epgu = 0
            self.amount_tickets_lpu = 0
            for row in self.payload:
                if row.post.lower() in self.ALLOWED_POSTS:
                    amount_tickets_in_row = row.total_for_epgu + row.total_for_lpu
                    if amount_tickets_in_row > 0:
                        self.data_by_lpu.setdefault(row.name, {})[row.post] = {
                            'for_epgu': row.total_for_epgu,
                            'busy_for_epgu': row.busy_for_epgu,
                            'for_lpu': row.total_for_lpu,
                            'amount': amount_tickets_in_row
                        }
                        self.amount_tickets += amount_tickets_in_row
                        self.amount_tickets_epgu += row.total_for_epgu
                        self.amount_tickets_lpu += row.total_for_lpu
            print('Получено организаций: ', len(self.data_by_lpu))
```

```
        def set_data_for_summary_gistogramm(self) -> None:
            """Собирает набор данных для основной гистограммы."""
            self.total_tickets_diagram_data = {}
            self.epgu_ticket_diagram_data = {}
            self.epgu_busy_ticket_diagram_data = {}
            for lpu_name, data in self.data_by_lpu.items():
                for item in data.values():
                    if lpu_name not in self.total_tickets_diagram_data:
                        self.total_tickets_diagram_data[lpu_name] = 0
                    if lpu_name not in self.epgu_ticket_diagram_data:
                        self.epgu_ticket_diagram_data[lpu_name] = 0
                    if lpu_name not in self.epgu_busy_ticket_diagram_data:
                        self.epgu_busy_ticket_diagram_data[lpu_name] = 0

                    self.total_tickets_diagram_data[lpu_name] += item['amount']
                    self.epgu_ticket_diagram_data[lpu_name] += item['for_epgu']
                    self.epgu_busy_ticket_diagram_data[lpu_name] += item['busy_for_epgu']

        def set_data_for_pie_diagrams(self) -> None:
            """Собирает набор данных для круговых диаграм."""
            self.pie_data = []
            for lpu_name in self.data_by_lpu:
                self.pie_data.append(
                    {
                        'name': lpu_name,
                        'epgu_percent': round(
                            self.epgu_ticket_diagram_data[lpu_name] * 100 /
self.total_tickets_diagram_data[lpu_name], 2
                        ),
                        'reverse_epgu_percent': round(
                            100 – self.epgu_ticket_diagram_data[lpu_name] * 100 /
self.total_tickets_diagram_data[lpu_name], 2
                        ),
                        'epgu_busy_percent': round(
                            self.epgu_busy_ticket_diagram_data[lpu_name] * 100 /
self.total_tickets_diagram_data[lpu_name], 2
                        ),
                        'reverse_epgu_busy_percent': round(
                            100 – self.epgu_busy_ticket_diagram_data[lpu_name] * 100 /
self.total_tickets_diagram_data[lpu_name], 2
                        ),
                    }
                )

        def read_payload_data(self, report_path: str) -> None:
            """Извлекает данные полезной нагрузки из excel отчета."""
            workbook = xlrd.open_workbook(report_path)
```

```
            sheet = workbook.sheet_by_index(0)
            for row_number in range(self.START_READ_ROW_NUMBER, sheet.nrows):
                self.payload.append(
                    ReportRow(
                        name=sheet.cell_value(row_number, 0),  # Название ЛПУ
                        post=sheet.cell_value(row_number, 2),  # Должность
                        total_for_epgu=int(sheet.cell_value(row_number, 12) or 0),  # Слотов для
ЕПГУ
                        total_for_lpu=int(sheet.cell_value(row_number, 13) or 0),  # Слотов для
ЛПУ
                        busy_for_epgu=int(sheet.cell_value(row_number, 19) or 0),  # Слотов для
ЕПГУ занято
                        busy_for_lpu=int(sheet.cell_value(row_number, 20) or 0),  # Слотов для
ЛПУ занято
                    )
                )
            print('Отчет прочитан.')
```

Приложение В

Программный код файла timetables.html

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <meta charset="utf-8" />
    <title>Графические отчеты</title>
    <style>
      html, body {padding:0; margin:0;}
      .content {
        background-color: white;
        display: block;
        min-height: 100vh;
        border: solid 1px #ecdcdc;
        margin-left: 2em;
        margin-right: 2em;
        padding-left: 2em;
        padding-right: 2em;
      }
      .post-list {
        display: flex;
        flex-wrap:wrap;
      }
      .post-list__item {
        display:block;
        position: relative;
        width: 20rem;
      }
    </style>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/chartjs-plugin-
datalabels/2.2.0/chartjs-plugin-datalabels.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-
ui/2.5.0/semantic.min.css" integrity="sha512-
KXol4x3sVoO+8ZsWPFI/r5KBVB/ssCGB5tsv2nVOKwLg33wTFP3fmnXa47FdSVIshVTgsYk/17
34xSk9aFIa4A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
  </head>
  <body>
    <div class="content">
      <h2>Статистика по биркам: {{ period }}</h2>
      <hr>
      <div class="ui info message">
        <div class="header">
          Специальности входящие в зачет:
        </div>
        <ul class="list post-list">
          {% for post_name in allowed_posts %}
            <li class="post-list__item">{{ post_name }}</li>
          {% endfor %}
```

```
            </ul>
        </div>

        <div class="ui statistics">
            <div class="statistic">
                <div class="value">{{ total }}</div>
                <div class="label">Всего создано слотов в РМИС</div>
            </div>
            <div class="statistic">
                <div class="value">{{ total_epgu }} ({{ total_epgu_percent }} %)</div>
                <div class="label">Доступно для ЕПГУ</div>
            </div>
            <div class="statistic">
                <div class="value">{{ total_lpu }}</div>
                <div class="label">Не доступно для ЕПГУ</div>
            </div>
        </div>

        {% for diagram in diagrams %}
            <h2>{{ diagram.group_name }}:</h2>
            <canvas id="ticketChart{{loop.index}}" width="600"
height="{{diagram.calc_height}}"></canvas>
            <hr>
            <script>
                var ticketCanvas{{loop.index}} =
document.getElementById("ticketChart{{loop.index}}");
                var ticketChart{{loop.index}} =
ticketCanvas{{loop.index}}.getContext("2d");
                var totalTicketsData{{loop.index}} = {
                    label: 'Всего слотов',
                    data: [{% for item in diagram.lpu_datasets %}{{ item.total_tickets_by_lpu
}},{% endfor %}],
                    backgroundColor: 'rgba(128, 99, 132, 0.6)',
                };
                var epguTicketsData{{loop.index}} = {
                    label: 'ЕПГУ создано',
                    data: [{% for item in diagram.lpu_datasets %}{{ item.epgu }},{% endfor
%}],
                    backgroundColor: 'rgba(0, 99, 132, 0.6)',
                };
                var epguPlanTicketsData{{loop.index}} = {
                    label: 'ЕПГУ План (60% от общего)',
                    data: [{% for item in diagram.lpu_datasets %}{{ item.epgu_norma }},{%
endfor %}],
                    backgroundColor: 'rgba(99, 132, 0, 0.6)',
                };
                var busyEpguTicketsData{{loop.index}} = {
```

```
                    label: 'ЕПГУ занято',
                    data: [{% for item in diagram.lpu_datasets %}{{ item.epgu_busy }},{%
endfor %}],
                    backgroundColor: 'rgba(0, 66, 300, 0.6)',
                };

                var barChart = new Chart(ticketChart{{loop.index}}, {
                    type: 'bar',
                    data: {
                        labels: [{% for item in diagram.lpu_datasets %}'{{ item.lpu_name
}}',{% endfor %}],
                        datasets: [totalTicketsData{{loop.index}},
epguPlanTicketsData{{loop.index}}, epguTicketsData{{loop.index}},
busyEpguTicketsData{{loop.index}}]
                    },
                    plugins: [ChartDataLabels],
                    options: {
                        indexAxis: 'y',
                        responsive: true,
                        plugins: {
                            datalabels: {
                            align: 'right',
                            anchor: 'end'
                            }
                        }
                    }
                });
            </script>
        {% endfor %}
        </div>
        <hr>
    </div>
</body>
</html>
```

Приложение Г

Информация о преобразовании из формата py в exe

Running auto-py-to-exe v2.44.2
Building directory: C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu
Provided command: pyinstaller --noconfirm --onefile --console "C:\Users\NizhalovNA\Desktop\mecm\dashboard_from_file\main.py"
Recursion Limit is set to 5000
Executing: pyinstaller --noconfirm --onefile --console C:\Users\NizhalovNA\Desktop\mecm\dashboard_from_file\main.py --distpath C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\application --workpath C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\build --specpath C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu

158404 INFO: PyInstaller: 6.11.0, contrib hooks: 2024.9
158411 INFO: Python: 3.10.11
158444 INFO: Platform: Windows-10-10.0.19043-SP0
158458 INFO: Python environment: C:\Users\NizhalovNA\AppData\Local\Programs\Python\Python310
158479 INFO: wrote C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\main.spec
158498 INFO: Module search paths (PYTHONPATH):
['C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\Scripts\\auto-py-to-exe.exe',

'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\python310.zip',
'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\DLLs',
'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib',
'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310',
'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages',
'C:\\Users\\NizhalovNA\\Desktop\\mecm\\dashboard_from_file']
158989 INFO: checking Analysis
158994 INFO: Building Analysis because Analysis-00.toc is non existent
158998 INFO: Running Analysis Analysis-00.toc
159010 INFO: Target bytecode optimization level: 0
159015 INFO: Initializing module dependency graph...
159034 INFO: Initializing module graph hook caches...
159074 INFO: Analyzing base_library.zip ...
159904 INFO: Processing standard module hook 'hook-encodings.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'
160496 INFO: Processing standard module hook 'hook-heapq.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'
160865 INFO: Processing standard module hook 'hook-pickle.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'
162707 INFO: Caching module dependency graph...
162858 INFO: Looking for Python shared library...

162885 INFO: Using Python shared library: C:\Users\NizhalovNA\AppData\Local\Programs\Python\Python310\python310.dll

162892 INFO: Analyzing C:\Users\NizhalovNA\Desktop\mecm\dashboard_from_file\main.py

163013 INFO: Processing standard module hook 'hook-pkg_resources.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

163021 INFO: SetuptoolsInfo: initializing cached setuptools info...

164937 INFO: Processing standard module hook 'hook-platform.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

164987 INFO: Processing standard module hook 'hook-xml.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

165043 INFO: Processing standard module hook 'hook-sysconfig.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

165427 INFO: Processing pre-safe-import-module hook 'hook-more_itertools.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165646 INFO: Processing pre-safe-import-module hook 'hook-jaraco.text.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165668 INFO: Processing standard module hook 'hook-jaraco.text.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages_pyinstaller_hooks_contrib\\stdhooks'

165836 INFO: Processing pre-safe-import-module hook 'hook-importlib_resources.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165840 INFO: Processing pre-safe-import-module hook 'hook-jaraco.context.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165854 INFO: Processing pre-safe-import-module hook 'hook-backports.tarfile.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165939 INFO: Processing standard module hook 'hook-backports.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages_pyinstaller_hooks_contrib\\stdhooks'

165963 INFO: Processing pre-safe-import-module hook 'hook-jaraco.functools.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

165985 INFO: Processing pre-safe-import-module hook 'hook-typing_extensions.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

166167 INFO: Processing pre-safe-import-module hook 'hook-importlib_metadata.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

166599 INFO: Processing pre-safe-import-module hook 'hook-six.moves.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

166956 INFO: Processing standard module hook 'hook-jinja2.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages_pyinstaller_hooks_contrib\\stdhooks'

167422 INFO: Processing standard module hook 'hook-multiprocessing.util.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

168179 INFO: Processing module hooks (post-graph stage)...

168202 INFO: Processing pre-safe-import-module hook 'hook-win32com.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages_pyinstaller_hooks_contrib\\pre_safe_import_module'

168985 INFO: Processing pre-safe-import-module hook 'hook-packaging.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\pre_safe_import_module'

169011 INFO: Processing standard module hook 'hook-packaging.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks'

169186 INFO: Performing binary vs. data reclassification (3 entries)

169209 INFO: Looking for ctypes DLLs

169307 INFO: Analyzing run-time hooks ...

169319 INFO: Including run-time hook 'pyi_rth_inspect.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\rthooks'

169335 INFO: Including run-time hook 'pyi_rth_pkgutil.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\rthooks'

169352 INFO: Including run-time hook 'pyi_rth_multiprocessing.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\rthooks'

169372 INFO: Including run-time hook 'pyi_rth_pkgres.py' from 'C:\\Users\\NizhalovNA\\AppData\\Local\\Programs\\Python\\Python310\\lib\\site-packages\\PyInstaller\\hooks\\rthooks'

169421 INFO: Looking for dynamic libraries

170314 INFO: Extra DLL search directories (AddDllDirectory): []

170317 INFO: Extra DLL search directories (PATH): []

170568 INFO: Warnings written to C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\build\main\warn-main.txt

170642 INFO: Graph cross-reference written to C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\build\main\xref-main.html

170697 INFO: checking PYZ

170703 INFO: Building PYZ because PYZ-00.toc is non existent

170719 INFO: Building PYZ (ZlibArchive) C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\build\main\PYZ-00.pyz

171390 INFO: Building PYZ (ZlibArchive) C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\build\main\PYZ-00.pyz completed

successfully.

171442 INFO: checking PKG

171450 INFO: Building PKG because PKG-00.toc is non existent

171465 INFO: Building PKG (CArchive) main.pkg

172978 INFO: Building PKG (CArchive) main.pkg completed successfully.

173000 INFO: Bootloader C:\Users\NizhalovNA\AppData\Local\Programs\Python\Python310\lib\site-packages\PyInstaller\bootloader\Windows-64bit-intel\run.exe

173011 INFO: checking EXE

173026 INFO: Building EXE because EXE-00.toc is non existent

173041 INFO: Building EXE from EXE-00.toc

173057 INFO: Copying bootloader EXE to C:\Users\NIZHAL~1\AppData\Local\Temp\tmp0md2l2gu\application\main.exe

173089 INFO: Copying icon to EXE

173111 INFO: Copying 0 resources to EXE

173119 INFO: Embedding manifest in EXE

173143 INFO: Appending PKG archive to EXE

173172 INFO: Fixing EXE headers

173403 INFO: Building EXE from EXE-00.toc completed successfully.

Moving project to: C:\Users\NizhalovNA\output

Complete.