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)

Разработка программного обеспечения
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка веб-приложения для анализа и визуализации данных»

Обучающийся Г.С. Редькин
(Инициалы Фамилия) (личная подпись)

Руководитель к.э.н., доцент, Т.А. Раченко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.ф.н., доцент, М.В. Дайнеко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Аннотация на дипломную работу студента Редькина Г.С. на тему: «Разработка веб-приложения для анализа и визуализации данных».

Работа включает в себя: 76 страниц, 8 таблиц, 60 рисунков, списком литературы в 20 позиций (включающий 7 иностранных), 0 приложений.

В работе исследовался процесс проведения анализа и визуализации данных в компании ООО «Ворлдинтертех», в ходе которого выявлены и проанализированы проблемы данного процесса. Для их решения была поставлена задача на проектирование и разработку единого программного решения, что поспособствует повышению эффективности процесса и нейтрализации его проблем.

В ходе работы было спроектировано веб-приложения для анализа и визуализации данных, и были реализованы его компоненты в виде модулей авторизации, механизмов обработки загружаемых пользователем CSV-файлов, визуализации и анализа данных, и также приняты меры по обеспечению его безопасности.

По окончанию реализации было проведено полномасштабное тестирование разработанных модулей, приведен контрольных пример и демонстрация путем фиксации последовательного пути пользователя приложения, а также проведен экономический расчет с показателем эффективности в 68,6%.

Abstract

The title of the graduation work is *Developing a web application for data analysis and visualization*.

The graduation work consists of an introduction, 3 parts, 60 figures, 8 tables, and a list of 20 references including 7 foreign sources.

The aim of this graduation work is to enhance the efficiency of the data analysis and visualization at OOO "WorldInterTech" (LLC under the laws of the Russian Federation) by designing and developing a unified web application.

The object of the graduation work is the data analysis and visualization process at the company in question.

The subject of the graduation work is the designed and developed web application for data analysis and visualization, including its authorization module, mechanisms for processing CSV files uploaded by the user, data visualization and analysis modules, as well as security features.

The key issue of the study is the identification and resolution of the problems related to the existing data analysis and visualization process at OOO "WorldInterTech" (LLC under the laws of the Russian Federation) by means of software development.

The graduation work may be divided into several logically connected parts which are: analysis of the existing process and problem identification; design of the web application solution; implementation of the key modules; testing and economic efficiency assessment.

The first part dwells on the data analysis and visualization process at the company under research, as well as identifies and examines its key problems.

The second part deals with designing the web application solution and implementing its main components: the authorization module, mechanisms for processing CSV files uploaded by users, data visualization and analysis modules, and security measures.

The third part is devoted to methodology of the developed modules complex testing, presents a test case and demonstrates the user journey. This part also focuses on the methodology of assessing the economic efficiency.

In conclusion, it should be highlighted that the comprehensive testing has confirmed the functionality of the developed modules. The economic efficiency assessment has demonstrated a positive result with a Return on Investment (ROI) indicator of 68.6%.

The work is of interest for a wide circle of specialists involved in business process optimization, software development for data analysis, and IT solution implementation in a corporate environment.

Содержание

Введение 6

1 Постановка задачи на разработку программного обеспечения для информационной системы предприятия 8

1.1 Описание деятельности и структуры предприятия. Значение программного обеспечения для информационной системы предприятия 8

1.2 Концептуальное моделирование предметной области 12

1.3 Анализ существующих разработок 14

1.4 Определение функциональных и нефункциональных требований 15

2 Процесс разработки программного обеспечения 20

2.1 Проектирование программного обеспечения 20

2.2 Выбор технологий разработки ПО 27

2.3 Реализация функциональных требований 29

2.3.1. Настройка проекта 29

2.3.2 Реализация модулей регистрации и авторизации 30

2.3.3 Реализация модуля загрузки CSV-файла 33

2.3.4 Разработка механизмов анализа данных и визуализация. 35

2.3.5 Обеспечение безопасности веб-приложения 41

2.3.6 Реализация пользовательского интерфейса 48

3 Оценка эффективности разработанного программного обеспечения 53

3.1 Тестирование функционала приложения 53

3.2 Тестирование безопасности 56

3.3 Контрольный пример и демонстрация 58

3.4 Оценка и обоснование экономической эффективности разработки ... 69

Заключение 73

Список используемой литературы и используемых источников 75

Введение

Разработка программного обеспечения для информационных систем предприятий прошла значительную эволюцию: от локальных решений с ограниченной функциональностью до комплексных платформ, интегрирующих анализ данных, управление ресурсами и искусственный интеллект. В 1990-х годах основой были ERP-системы, фокусирующиеся на автоматизации внутренних процессов. С развитием облачных технологий и Big Data в 2010-х акцент сместился на гибкость, масштабируемость и аналитику в реальном времени. Сегодня ключевыми трендами стали низкокодовые платформы [1], микросервисная архитектура [3, 7] и интеграция машинного обучения, что позволяет предприятиям адаптироваться к динамичным рыночным условиям.

Актуальность темы обусловлена растущей потребностью предприятий в автоматизации и оптимизации рутинных операций и повышении точности аналитики. По данным исследования Gartner [5], 78% компаний сталкиваются с потерями из-за ошибок при ручной обработке данных. Для ООО "Ворлдинтертех РУС", чья деятельность включает разработку ПО и управление проектами, внедрение специализированного решения позволит сократить затраты и ускорить принятие решений.

Существующие исследования, такие как работы А.И. Петрова по оптимизации бизнес-процессов [10], и И.С. Козлова по проектам в области веб-аналитики [11], подтверждают эффективность автоматизации. Однако большинство решений не учитывают специфику малых и средних предприятий, что ограничивает их применимость. Данная работа восполняет этот пробел, предлагая модульное ПО, которое представляет необходимый функционал и которое легко поддерживать.
Цель работы - разработать веб-приложение, предоставляющее функционал анализа и визуализации над данными.

На пути достижения данной цели необходимо было решить следующий спектр задач:

- Смоделировать проблемный процесс и проанализировать причины его неэффективности;

- Спроектировать модель целевого состояния процесса;
- Сформулировать требования к функционалу и архитектуре;
- Спроектировать модульную структуру ПО;
- Реализовать ключевые функции (анализ данных, визуализация, интеграция с СУБД);
- Провести тестирование на соответствие требованиям;
- Провести расчет экономической эффективности.

Объектом работы является процесс разработки веб-приложений для анализа данных и визуализации.

Предметом выступают методы, инструменты, технологии проектирования, обработки, анализа и визуализации данных.

Новизна работы заключается в адаптации микросервисной архитектуры под нужды малого бизнеса. Практическая ценность подтверждена расчетом экономической эффективности: внедрение ПО сократит время анализа данных на 68%, а срок окупаемости составит 104 дня. Результаты могут быть применены в IT-компаниях, логистике и розничной торговле.
Работа состоит из трех глав:

- Постановка задачи: Анализ проблем, требований и существующих решений;
- Проектирование и реализация: Описание архитектуры, выбора технологий и этапов разработки;
- Тестирование и оценка: Всестороннее тестирование, контрольный пример, экономический расчет.

Во введении обозначены ключевые аспекты исследования, которые будут детализированы в последующих разделах.

1 Постановка задачи на разработку программного обеспечения для информационной системы предприятия

1.1 Описание деятельности и структуры предприятия. Значение программного обеспечения для информационной системы предприятия

ООО «Ворлдинтертех РУС» является молодой коммерческой организацией с основным видом деятельности - Разработка компьютерного программного обеспечения (код по ОКВЭД 62.01). Дополнительно организация занимается следующими видами деятельности:

- 33.20 Монтаж промышленных машин и оборудования;
- 46.51 Торговля оптовая компьютерами, периферийными устройствами к компьютерам и программным обеспечением;
- 47.41 Торговля розничная компьютерами, периферийными устройствами к ним и программным обеспечением в специализированных магазинах;
- 62.03 Деятельность по управлению компьютерным оборудованием;
- 62.09 Деятельность, связанная с использованием вычислительной техники и информационных технологий.

В реализации основного вида деятельности организации лежит выполнение обязательств по договорам на разработку приложение и их внедрения, а также разработка собственных проектов компании и их сопровождение.

Отдел, который непосредственно выполняет всю данную работу – отдел разработки и проектирования ПО. Структура организации показана на рисунке 1.

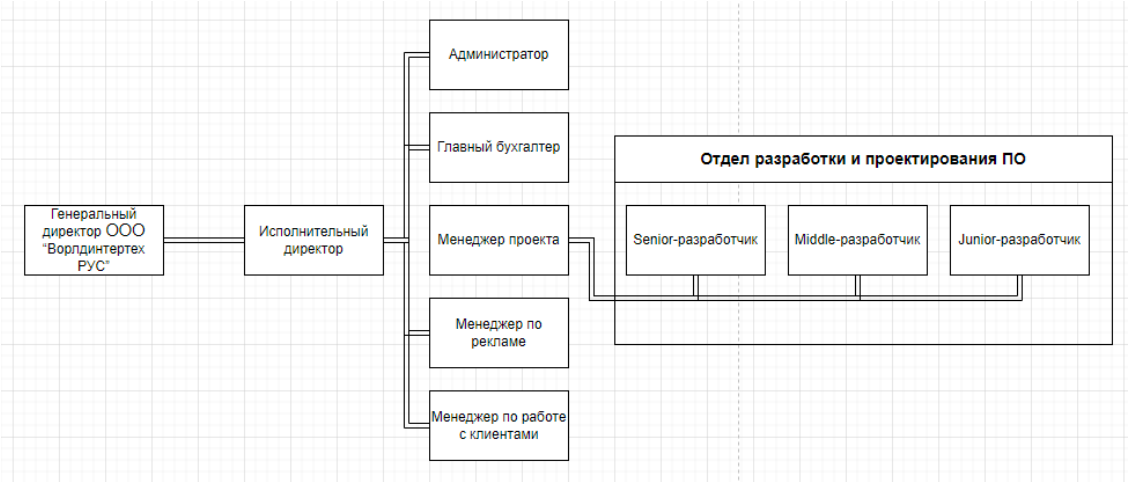


Рисунок 1 – Организационная структура ООО «Ворлдинтертех РУС»

Как видно из Рисунка 1, ключевым подразделением, чья деятельность завязана на разработке и проектировании ПО, является отдел разработчиков. Бизнес-процесс разработки ПО запускается в момент заключение договора на разработку между компанией и заказчиком. Команда отдела разработки принимает участие в реализации каждого подпроцесса, использую для работы соответствующие средства разработки. Все подпроцессы реализуются строго поочередно, и выполняются в соответствии с условленными требования заказчика, содержащимися в техническом задании, а также регламенту самой организации.

По заключению договора на разработку – команда разработчиков начинает проектирование ПО, в результате чего прорабатывается архитектура системы и ее дизайн. На основе спроектированных данных, происходит реализация компонентов системы и пишется ее исходный код на условленном языке программирования. Как написание кода программы завершается, компоненты программы подвергаются тщательному тестированию и испытаниям, по итогу чего на выходе имеем отчеты о работе программы и ее ошибках. С помощью этих данных, в следующем этапе происходит отладка программы и исправление найденных багов. Как только отладка завершается и приложение полностью протестировано и соответствует заявленным заказчиком требования – команда разработчиков помогает внедрить ее в систему клиента, по окончанию чего имеем готовое ПО и документацию по нему.

В процессе разработки и внедрения программного обеспечения, а также в других видах деятельности компании, таких как торговля компьютерами и программным обеспечением, управление компьютерным оборудованием и монтаж промышленных машин, анализ данных играет ключевую роль. Эффективное использование данных позволяет компании принимать обоснованные решения, оптимизировать бизнес-процессы и повышать конкурентоспособность на рынке.

Однако в текущей практике ООО «Ворлдинтертех РУС» анализ данных и их визуализация выполняются неэффективно и затрачивают очень много времени. Для наглядного отражения причин, была применена диаграмма Исикавы (рисунок 2):

Рисунок 2 – Диаграмма Исикавы «Причина-следствие»

Применения диаграммы Исикавы помогло систематизировать причины неэффективности текущего состояния бизнес-процесса, выделив четыре основных группы факторов:

– Человеческий фактор: Сюда относятся такие причины, как ошибки при вводе данных и недостаточная квалификация работников. Это самый наименее оказывающий влияние на появлении текущий фактор, так как в организации автоматически производится сбор исходных данных для анализа и ошибки тут редки. А недостаточная квалификация указывается в качестве причины в силу того, что как используется в

процессе анализа сразу несколько инструментов и программ, а так как в своей работе производиться анализ данных время от времени приходится многим работников, то оказывается, что не у всех есть опыт работы с теми или иными инструментами, что по итогу может являться причиной ошибок в аналитике и увеличению длительности данного процесса.

- Методы работы: По причинам того, что нет единого инструментария, работниками приходится пользоваться несколькими и некоторые предполагают ручной ввод параметров визуализации и анализа, что повышает количество времени, затраченного на все эти процессы. А в следствии отсутствия стандартизации, результаты анализ и визуализаций одних данных могут разниться и хаотично храниться после на устройствах, и при надобности придется выделять дополнительно время на их поиск и систематизацию, или в худшем случае – повторять процесс. Данный фактор является самым ключевым и влиятельным.
- Организационные факторы: В связи с высокой стоимостью на профессиональные решения в данной области и их внедрения, а также нехватки времени на полноценных анализ – так же являются первопричинами проблемы неэффективного анализа и того, почему еще не был осуществлен переход с текущих решений на более эффективные.
- Технологические ограничения: Из-за того, что отсутствует интеграция между инструментами – то в случаях необходимости проведения многоэтапного анализа или визуализации, приходится меняться между инструментами и загружать в каждый данные по несколько раз, что так же является причиной неэффективности. И если приходится задействовать в этих процессах большие объемы данных – в их обработке будут иметься задержки и при условиях того, что еще может понадобиться пользоваться разными инструментария и проводить несколько типов анализов и визуализации – задержки будут

увеличиваться кратно их количеству. Данный фактор является так же значимой причиной неэффективности.

1.2 Концептуальное моделирование предметной области

В описание бизнес-процесса анализа данных и визуализации было выяснено, что они являются неоптимизированным из-за факторов, чтобы были выделены на диаграмме Исикавы (Рисунок 2). Для наглядности и выяснения, где конкретно какой фактор становится причиной неэффективности по ходу выполнения процесса. Для этого данный процесс был смоделирован в варианте «КАК-ЕСТЬ» в нотации IDEF0 (рисунок 3):

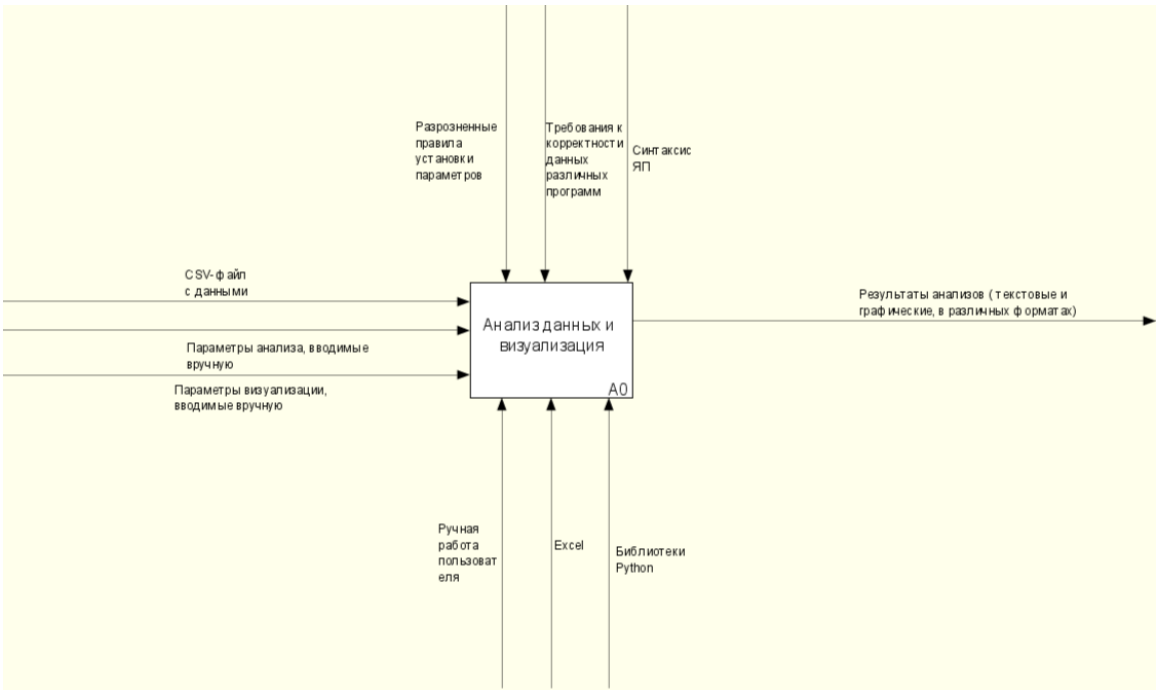


Рисунок 3 – Основной блок бизнес-процесса «Анализ данных и визуализация» в нотации IDEF0 в варианте «КАК-ЕСТЬ»

Входными данными здесь являются CSV-файл, в которых в организации преимущественно собираются предварительно данные, и параметры анализа и визуализации, что вносятся вручную в программном виде и непосредственным взаимодействием с инструментами. Управляется данный процесс разрозненно, так как в нету каких-либо общих правил в организации по проведению анализов и визуализаций различного типа, а также хранения их результатов. Также к

регуляторам еще относятся требования к корректности данных в используемых программах и синтаксис языка программирования, которым в основном является Python. В качестве механизмов, двигающих этот процесс, является ручная работа сотрудника, он же пользователь, а также сами внутренние инструменты Excel и функционал, предоставляемый библиотеками Python.

На рисунке 4 представлена декомпозиция основного блока текущего варианта проведения процесса.

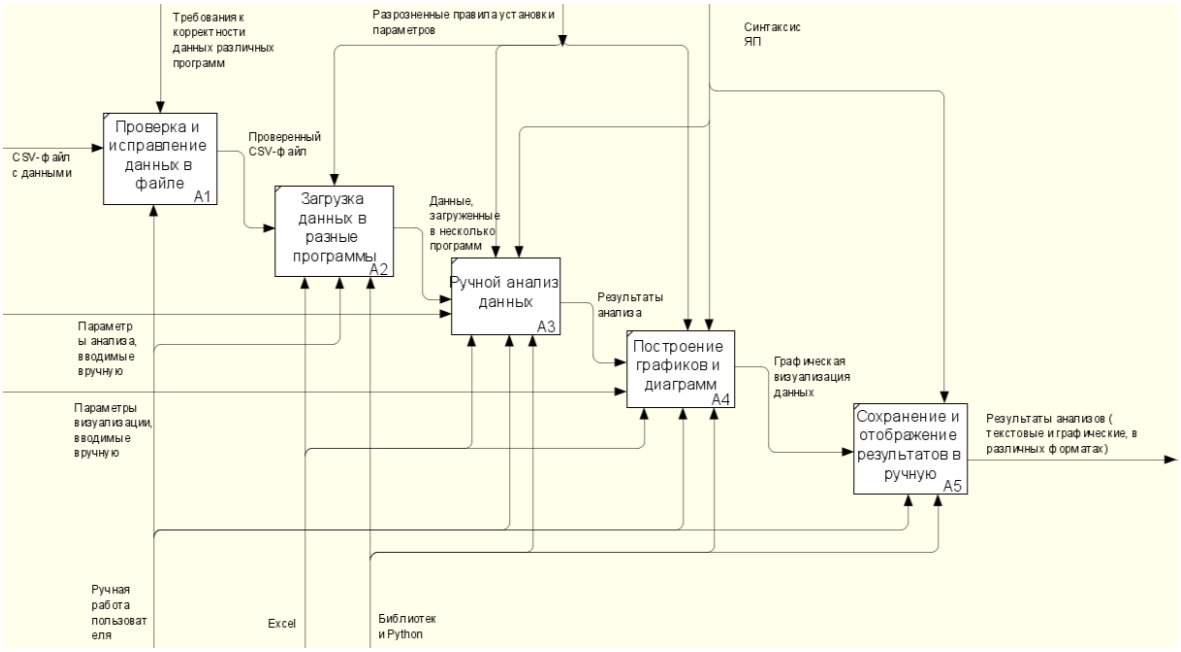


Рисунок 4 – Декомпозиция основного блока бизнес-процесса «Анализ данных и визуализация» в нотации IDEF0 в варианте «КАК-ЕСТЬ»

В текущем состоянии бизнес-процесса можно выделить следующие проблемы и отрицательные факторы:

– Разрозненность инструментов: пользователю приходится переключаться между программами и загружать в них данные, что занимает достаточно много времени само по себе;
– Обильность ручных задач: данная проблема усугубляет предыдущую, потому что помимо того, что вручную приходится переносить данные из программы в программу, так еще в каждой программе вручную и в соответствии с местным синтаксисом и правилами задачи параметров;

– Ошибки при переносе данных: в процессе постоянных импортов и экспортов между программы, фрагменты данных могут быть утеряны или искажены;
– Неоптимальное хранение результатов: при сохранении результатов и их обмене есть риск их потери и сложности организации общего доступа к файлам при их увеличении из-за отсутствия систематизации.

Наличие данных проблем делает уменьшает эффективность процесса, при том он является достаточно важным процессом в экосистеме организации. В связи с этим, по инициативе студента, было принято решение запуска проекта по разработке веб-приложения для оптимизации и уменьшении затрачиваемого времени на выполнение бизнес-процесса.

1.3 Анализ существующих разработок

На основе выделенных в прошлом шаге недостатков текущего положения дел и отрицательных аспектов проведения в организации процесса анализа и визуализации, проведем анализ существующих разработок-аналогов, на предмет возможности решения текущих проблем своим функционалом и особенностями.

Результаты сравнения аналогов приведены в таблице 1.

Таблица 1 – Сравнение аналогов

Требования/Аналог	Tableau	Microsoft Power BI	Google Data Studio	Проектируемое приложение
Поддержка анализа данных	+	+	+	+
Гибкость визуализации	+	+	-	+
Интеграция с CSV	+	+	+	+
Сохранение результатов	+	+	+	+
Поддержка статистического анализа	-	-	-	+

Продолжение таблицы 1

Требования/Аналог	Tableau	Microsoft Power BI	Google Data Studio	Проектируемое приложение
Личный кабинет с хранением и возможностью скачивания результатов в любой момент	-	-	-	+
Удобство использования	+	+	+	+
Доступность	-	-	-	+
Стоимость	Высокая	Средняя	Бесплатно	Только траты на разработку и поддержку
Количество положительных аспектов:	5	5	3	6

Существующие решения не полностью соответствуют к требованиям, особенно в части статистического анализа и удобства хранения результатов. В то время как проектируемое приложение предлагает уникальные функции и может быть тонко настроено и разработано исходя из требований оптимизации данного процесса в организации, в связи с чем решение разработки является вполне обоснованным и целесообразным.

1.4 Определение функциональных и нефункциональных требований

Для того чтобы выделить требования к проектируемому программному обеспечению и далее выбрать стек используемых технологий, необходимо предварительно построить модель «КАК-ДОЛЖНО-БЫТЬ».

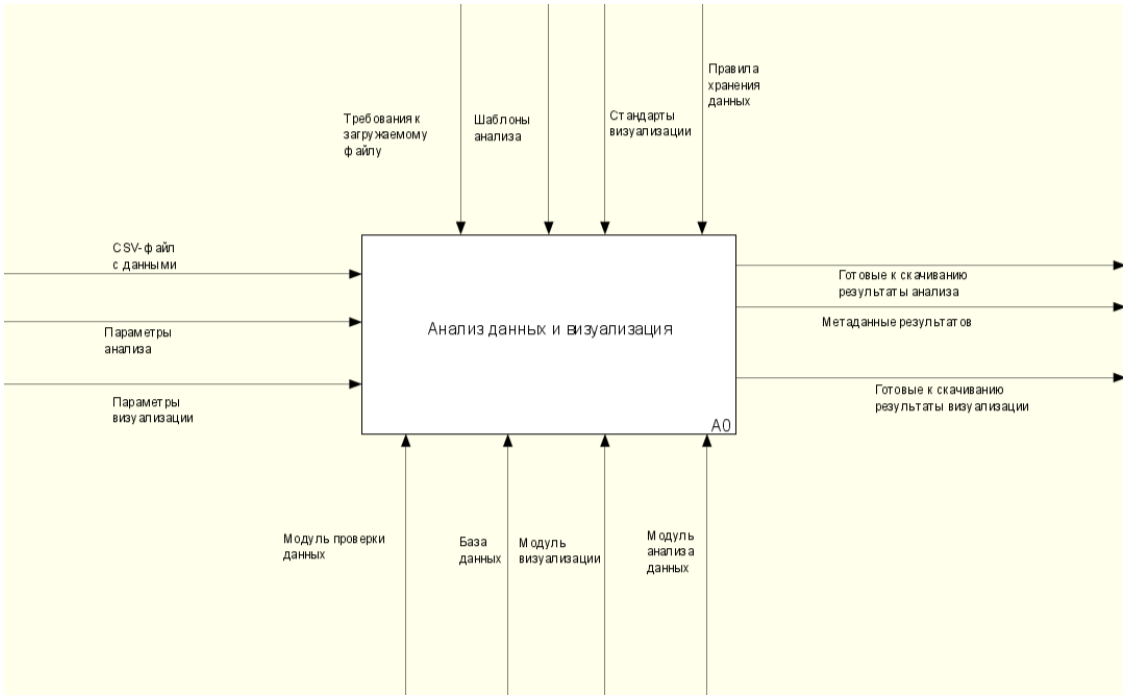


Рисунок 5 – Основной блок диаграммы варианта процесса «КАК-ДОЛЖНО-БЫТЬ» в нотации IDEF0

Разработанная модель целевого состояния процесса (рисунок 5) иллюстрирует концепцию единого веб-приложения, покрывая проблемы текущего состояния процесса. На ней оптимизированы все этапы: от загрузки и проверки данных до анализа, визуализации, сохранения результатов и предоставления доступа к ним через личный кабинет. Модель варианта процесса «КАК-ДОЛЖНО-БЫТЬ» стала основной для выделения требований к разрабатываемой системе.

На рисунке 6 представлена декомпозиция основного блока варианта процесса «КАК-ДОЛЖНО-БЫТЬ».

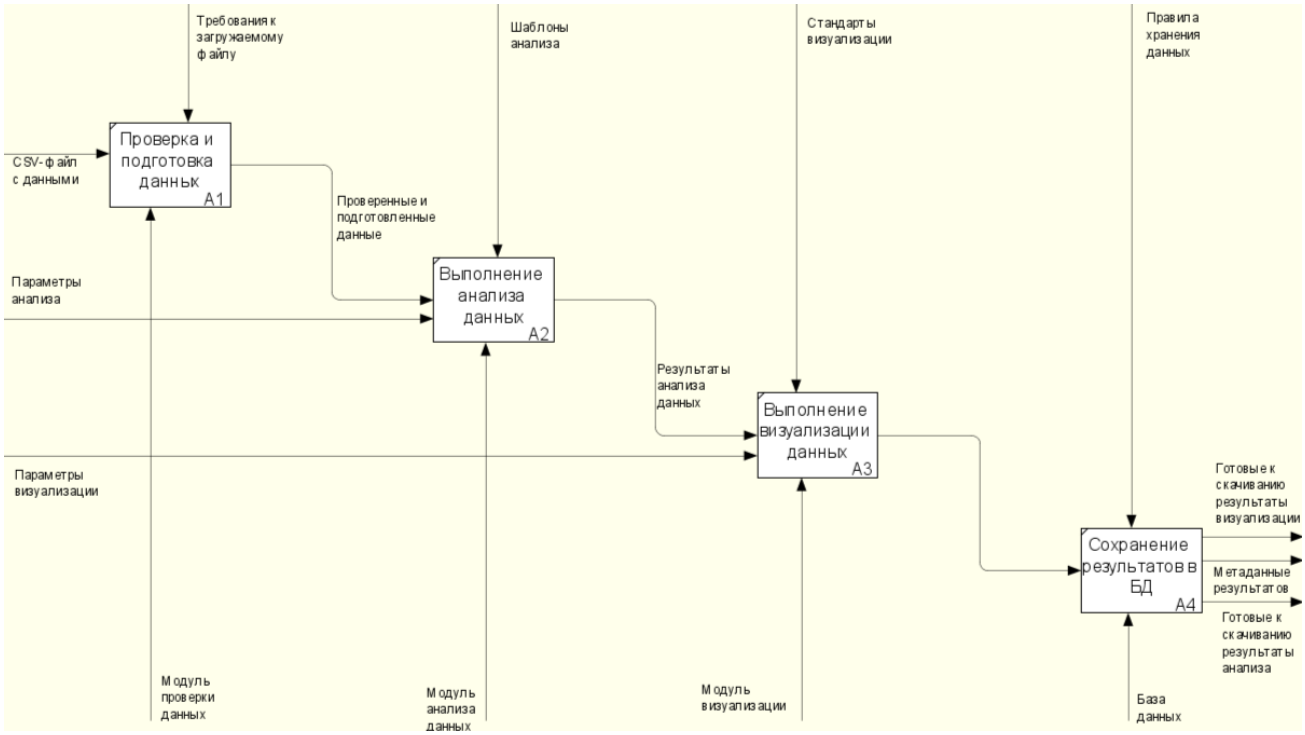


Рисунок 6 – Декомпозиция основного блока варианта процесса «КАК-ДОЛЖНО-БЫТЬ» в нотации IDEF0

В отличии от модели исследуемого процесса «КАК-ЕСТЬ», на данной все этапы выполняются в одном приложении, а содержание файла проверяется автоматически и в случае чего, конкретно указывает пользователю на несоответствия. Анализ данных и визуализации выполняется по шаблонам, и от пользователя лишь требуется выбрать желаемые настройки, всем остальным займется программа. Она отобразит результат пользователю, предоставит ему возможность его скачать сразу, и на случай чего сохранит метаданные результатов в базе данных, для доступа в любое время в личном кабинете. Единые стандарты анализа и визуализации обеспечат согласованность результатов. Все выделенные аспекты делают данный процесс куда более надежным, и что не менее важно – менее затратным в плане времени.

На основе модели процесса «КАК-ДОЛЖНО-БЫТЬ» и специфики предприятия, определим требования к проектируемому ПО в таблице 2 по методологии FURPS+ [4].

Таблица 2 – Требования к приложению по методологии FURPS+

Требование	Статус	Полезность	Риск
Функциональные требования			
Регистрация и авторизация пользователей.	Реализовано	Обеспечивает безопасный доступ к системе.	Риск утечки данных при слабой защите.
Загрузка CSV-файлов с автоматической проверкой формата и данных.	Реализовано	Упрощает процесс загрузки данных, минимизирует ошибки.	Риск некорректной обработки файлов при ошибках в алгоритме проверки.
Выполнение анализа данных	Реализовано	Позволяет быстро получать результаты анализа.	Риск некорректных результатов при ошибках в алгоритмах анализа.
Генерация визуализации на основе шаблонов.	Реализовано	Упрощает создание графиков, обеспечивает согласованность.	Риск неудовлетворительной визуализации при неправильных настройках.
Сохранение результатов и метаданных в базе данных.	Реализовано	Обеспечивает долгосрочный доступ к результатам.	Риск потери данных при сбоях в БД.
Удобство использования			
Интуитивно понятный интерфейс для загрузки данных, анализа и визуализации.	Реализовано	Упрощает работу пользователей, снижает время обучения.	Риск неудобства при непродуманном дизайне интерфейса.
Возможность настройки параметров анализа и визуализации.	Реализовано	Позволяет адаптировать систему под нужды пользователей.	Риск перегруженности интерфейса настройками.
Надежность			
Автоматическая проверка данных на корректность.	Реализовано	Минимизирует ошибки, связанные с некорректными данными.	Риск ложных срабатываний проверки.
Резервное копирование результатов и метаданных.	Реализовано	Обеспечивает сохранность данных при сбоях.	Риск потери данных при отсутствии резервного копирования.
Производительность			
Быстрая обработка данных (анализ и визуализация).	Реализовано	Увеличивает скорость работы пользователей.	Риск замедления при больших объемах данных.
Поддержка работы с большими объемами данных (до 16 мб на текущий момент)	Реализовано	Позволяет обрабатывать крупные файлы.	Риск сбоев при недостаточной оптимизации.
Поддерживаемость			
Модульная архитектура для легкого добавления новых функций.	Реализовано	Упрощает развитие и поддержку системы.	Риск увеличения сложности кода при плохой архитектуре.

Продолжение таблицы 2

Требование	Статус	Полезность	Риск
Ограничения			
Требование к интернет-соединению для работы с веб-приложением.	Реализовано	Упрощает доступ к системе из любого места.	Риск недоступности при отсутствии интернета.
Ограничение на размер загружаемого файла (до 16 мб на текущий момент).	Реализовано	Защищает систему от перегрузки.	Риск неудовлетворенности пользователей с большими файлами.
Поддержка только CSV-формата для загрузки данных.	Реализовано	Упрощает реализацию, фокусируясь на одном формате.	Риск неудобства для пользователей, работающих с другими форматами.

Таблица 2 послужит основой для планирования разработки приложения и в целом является полезным инструментом для управления требованиями и обеспечением успешной реализации проекта.

Вывод по главе 1.

В первой главе выпускной квалификационной работы была описана деятельность предприятия ООО «Ворлдинтертех РУС» с выделением и анализом проблемного бизнес-процесса. В рамках анализа процесс анализа и визуализации данных был смоделирован в нотации IDEF0 в текущем варианте его исполнения в описанной выше организации. По результатам анализа были выявлены основные причины его неэффективности, отталкиваясь от которых была спроектирована модель данного процесса в вариации «КАК-ДОЛЖНО-БЫТЬ».

На основе спроектированной модели, был проведен поиск и анализ существующих программных решений, способных оптимизировать процесс. По итогам анализа, наибольшее количество положительных аспектов имелось у решения с разработкой собственного веб-приложения для предприятия, так как помимо перспективы оптимизации процесса визуализации и анализа данных в компании, будет учитываться специфика рассматриваемого предприятия.

На основе модели целевого варианта проведения процесса, к разрабатываемому ПО были выдели требования по категориям, следуя методологии FURPS+.

2 Процесс разработки программного обеспечения

2.1 Проектирование программного обеспечения

На основе выделенных требования, лучшим вариантом для данного проекта, будет использование трехзвенной архитектуры (клиент-серверной), что показана на рисунке 7.

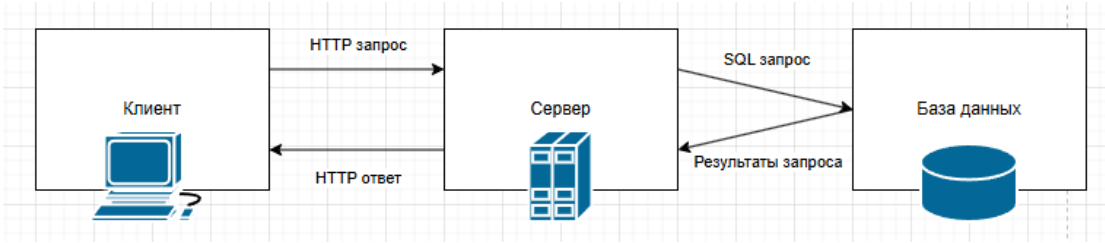


Рисунок 7 – Клиент-серверная архитектура приложения

Так будут разделена бизнес-логика и данные, а также такую систему удобно поддерживать и масштабировать, в связи с чем ее выбор обоснованно является оптимальным [18].

На основе выделенных к системе функциональных требований, составлена диаграмма вариантов пользования (рисунок 8).

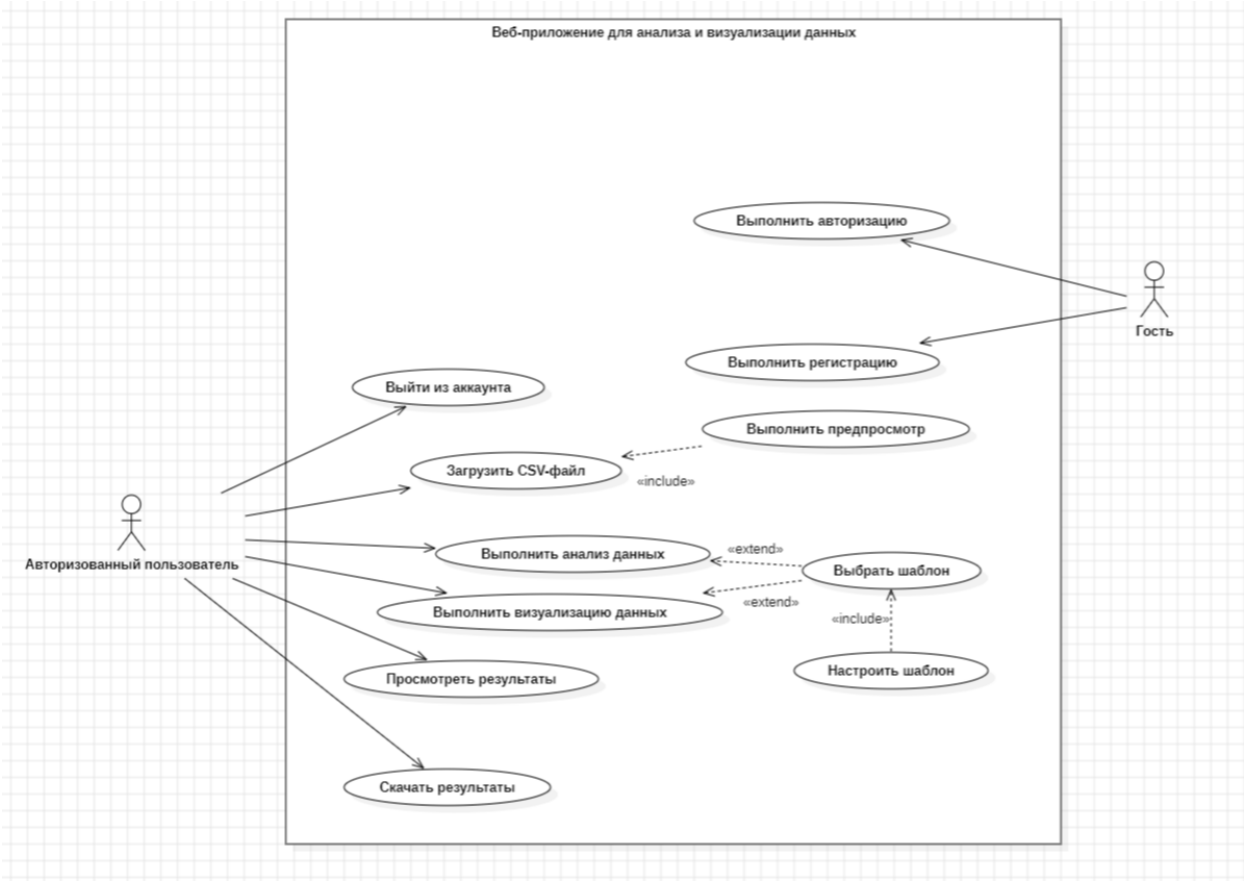


Рисунок 8 – Диаграмма вариантов пользований с отображением основных компонентов.

На диаграмме вариантов пользования были определены роли пользователей, и ключевые функции, что будут предоставляться в соответствии с ней системой.

На ней представлены два актора:

- Гость (пользователь, не вошедший в систему) – ему доступны лишь два прецедента, и полноценный функционал приложения ему недоступен;
- Авторизованный пользователь – имеет в доступе еще больше прецедентов, в связи с чем может пользоваться всем функционалом, предоставляемым программой.

Разделение ролей необходимо в связи с планами реализации личного кабинета, для которого нужно авторизоваться в системе. Помимо этого, так как функции приложения предполагают загрузку пользовательских данных в CSV-формате и результаты будут строиться по ним – необходимо и обеспечить возможность доступа лишь соответствующего пользователя к ним в рамках обеспечения безопасности.

На основе спроектированной диаграммы был разобран и задокументирован каждый прецедент в таблице 3.

Таблица 3 – Документирование объектов диаграммы вариантов пользования

Название прецедента	**Актор**	**Описание**
Загрузить CSV-файл	Авторизованный пользователь	Авторизованный пользователь может открыть форму загрузки файла и загрузить в веб-приложение свой CSV-файл
Выполнить авторизацию в системе	Гость	Пользователь может выполнить вход в систему, используя данные, которые использовались им при регистрации
Скачать результаты	Авторизованный пользователь	Пользователь может скачать результаты анализа и визуализации в форматах PNG или HTML.
Выполнить регистрацию в системе	Гость	Гость может зарегистрироваться в системе, указав логин, email и дважды пароль.
Выбрать шаблон анализа	Пользователь	Пользователь может выбрать тип анализа (описательная статистика, корреляция, группировка).
Выбрать шаблон	Авторизованный пользователь	Пользователь может выбрать тип визуализации (гистограмма, круговая диаграмма и т.д.) или выбрать тип анализа (описательная статистика, корреляция, группировка).
Настроить параметры шаблона	Авторизованный пользователь	Пользователь может настроить параметры выбранного шаблона анализа или визуализации.
Выйти из аккаунта	Авторизованный пользователь	Пользователь может завершить сеанс работы с системой.
Просмотреть результаты	Авторизованный пользователь	Пользователь может просмотреть результаты выполненных по шаблону анализов или визуализаций.
Выполнить анализ данных	Авторизованный пользователь	Авторизованный пользователь посредством функционала программы выполняет анализ данных в соответствии с выбранным шаблоном.
Выполнение визуализации данных	Авторизованный пользователь	Авторизованный пользователь посредством функционала программы графики и диаграммы.

Документирование каждого прецедента формализовало взаимодействие пользователя с системой для выполнения основных задач, в связи чем можно переходить к следующим этапам проектирования.

Далее спроектируем модель компонентов системы. Определимся с их расположением в ней и составляющих ее модули. Так как была выбрана клиент-серверная архитектура – в построении будем отталкиваться от нее и так же от

того, что где-то в таком случае будет необходимо хранить пользовательские данные и данные о результатах, созданными ими.

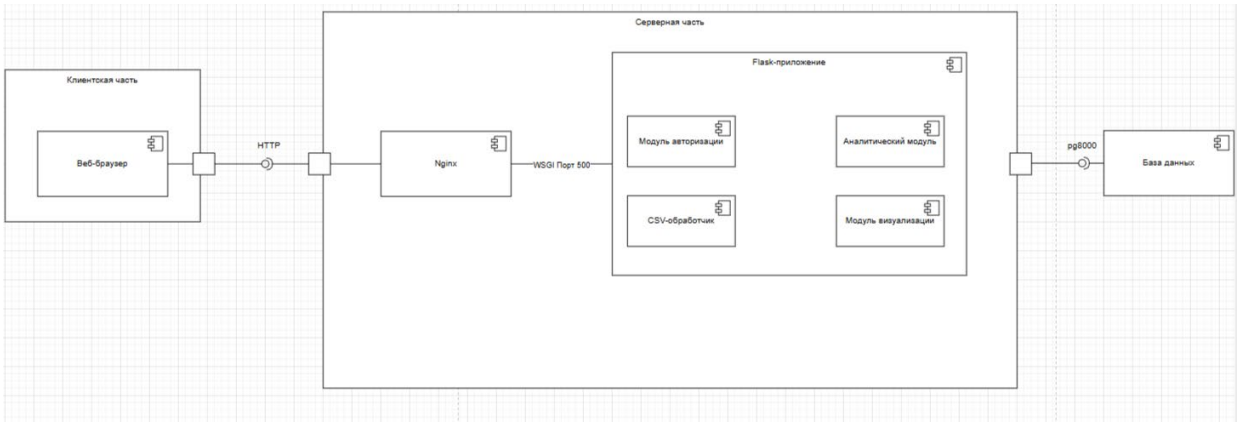


Рисунок 9 – Диаграмма компонентов

Информационная система представляет на диаграмме компонентов на рисунке 9 собой три связанные части:

- Клиентская часть: здесь компонентом представлен веб-браузер, что рендерит для пользователя веб-страницы и с которым пользователь, взаимодействуя, отправляет запросы на сервер;
- Серверная часть: компонент Nginx, который будет заниматься обработкой статики и перенаправлением API-запросов на Flask. И само приложением Flask – основной компонент системы, так же относится к этой части. В себя включает четыре самостоятельных модуля, каждый из которых выполняет соответствующую его названию функцию;
- База данных: связанный с серверной частью компонент, с которым она взаимодействует с помощью драйвера, что предполагается будет хранить информацию о пользователях и результатах.

Так как целью работы является разработка веб-приложения, необходимо было и продумать веб-страницы, что будут рендериться на пользовательском устройстве. В таблице 4 далее будут расписаны типы страниц и пояснение по ним.

Таблица 4 – Проектирование веб-страниц приложения

Тип страницы	Доступность	Описание
Главная страница	Всем	Главная страница приложения, в будущем может использоваться для размещения на себе различной информации. На момент разработки планируется содержать кнопку перехода на страницу загрузки файла CSV. Для гостя дополнительно будет размещена ссылка с предложением авторизоваться для доступа к функционалу
Страница авторизации	Всем	Страница с формой, где пользователь может ввести свои данные учетные записи и войти в систему под именем аккаунта.
Страница регистрации.	Всем	Страница с формой, где пользователи, не имеющие аккаунт в системе могут его задать с необходимыми в форме учетными данными
Страница выгрузки CSV-файла	Только авторизованным пользователям	Страница, где авторизованные пользователи могут загрузить в систему CSV с файл со своими данными, для последующего анализа или визуализации
Страница предпросмотра	Только авторизованным пользователям	Странице, на которой пользователь отображаться информация о файле и, в случае наличия пустых ячеек в файле, будет уведомление о них и выбор, как с ними программе поступать.
Страница настройки	Только авторизованным пользователям	Страница с двумя разделами – визуализации и анализа, между которыми можно переключаться, где будут предоставлены соответствующие шаблоны и их настройки.
Страница просмотра результатов анализа	Только авторизованным пользователям	Страница с отображением пользователю результатов по выбранному им типу анализа и кнопками для скачивания.
Страница просмотра результатов визуализации.	Только авторизованным пользователям	Страница с отображением пользователю результатов по выбранному им типу визуализации и кнопками для скачивания.
Страница личного кабинета	Только авторизованным пользователям	Страница, где будет отображаться таблица с последними результатами, созданными пользователем.

Для реализации указанных страниц в таблице выше, требуется четкое разделение компонентов, что было уже спроектировано и показано на диаграмме компонентов. Теперь необходимо конкретизировать компоненты и их взаимодействие в рабочей среде, что отразим в процессе построения диаграммы развертывания на рисунке 10.

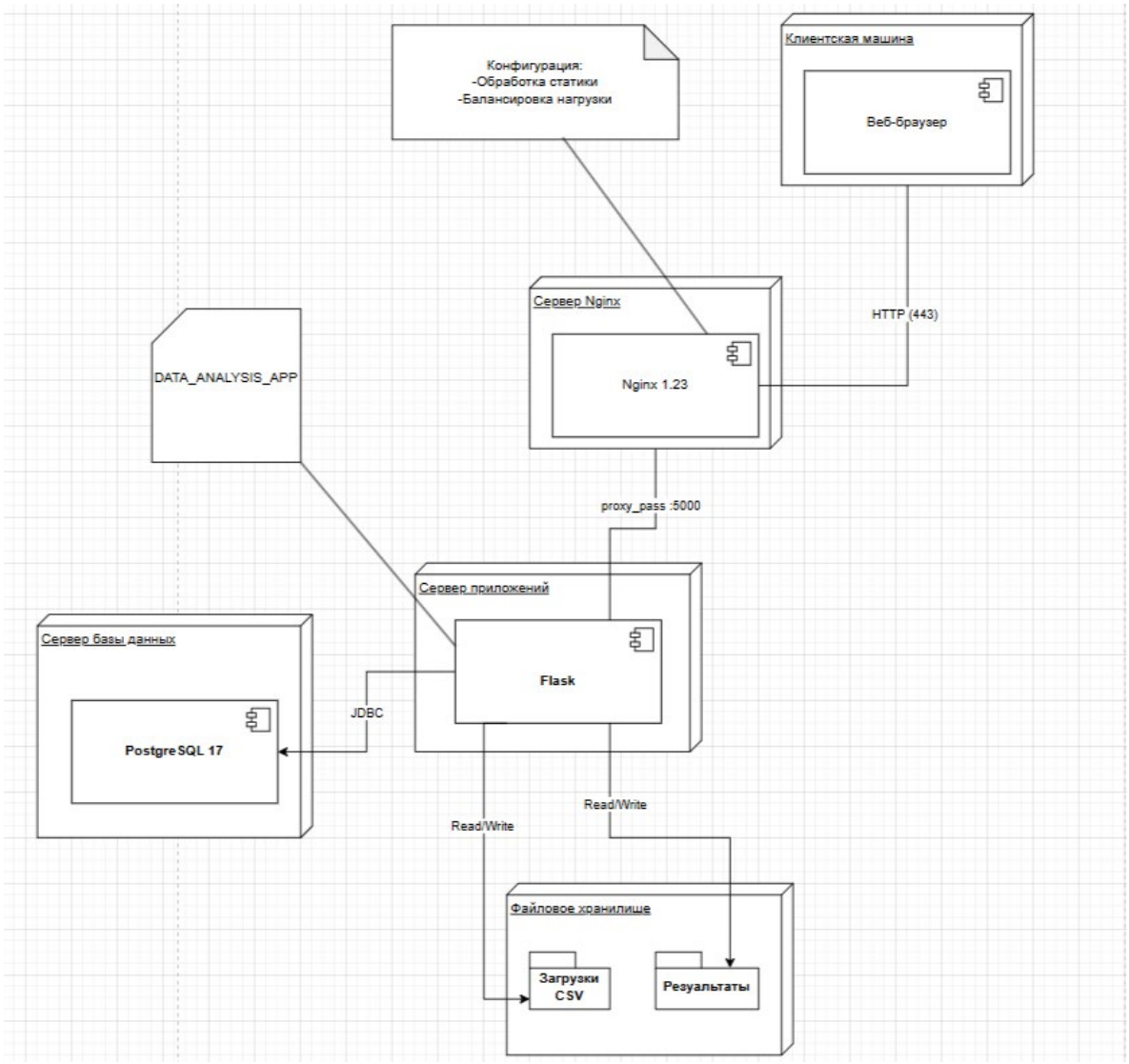


Рисунок 10 – Диаграмма развертывания

На диаграмме развертывания отобразили компоненты системы, а также связи между ними. Клиентская машина связывается с сервером Nginx посредством HTTP, который в свою очередь является промежуточным в связи между клиентом и основным приложением. Серверу Nginx задана конфигурации, в соответствии с которой клиенту рендерится статика и балансируется нагрузка. Nginx проверяет так же в соответствии со своей конфигурацией отправленные зашифрованные пользовательские запросы на 443 и перенаправляет их с помощью механизма proxy_pass к порту основного приложения. В зависимости от типа запроса, приложением задействуется соответствующий ему модуль. Если это запрос на авторизацию и регистрацию, с помощью драйвера pg8000 сервер отправляет параметризованный запрос в PostgreSQL на проверку наличия пользователя в БД или занесение его данных

туда соответственно [14]. А если запрос на проведение операций или загрузку своего CSV файла, то происходит взаимодействия приложения с файловым хранилищем, куда в папку каждого пользователя сохраняются его результаты, а в папку CSV-файлов – загружаемые именно пользователями файлы, что хранятся временно на момент, пока пользователь используется функционал приложения.

На основе диаграммы прецедентов спроектируем вид базы данных проекта. Необходимо создать две таблицы-модели для результата и пользователя. Физическая модель базы данных на рисунке 11 определяет структуру таблиц Пользователей и Результатов.

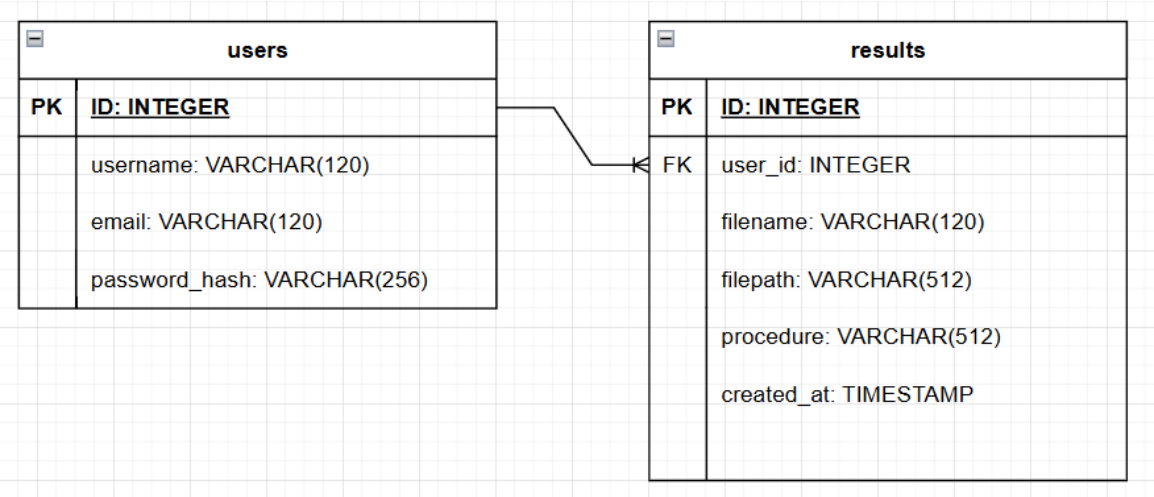


Рисунок 11 – Физическая модель БД

Таблица пользователя будет хранить ID, логин, почту и хэш пароля пользователя. ID пользователя в данной таблице будет являться первичным ключом, и будет соединен со своим внешним эквивалентом в таблице результата. Так как подразумевается, что у одного пользователя может храниться множество результатов анализов и визуализации – связь между таблицами будет «один-ко-многим». Таблица результата будет хранить в себе метаданные файлов результатов с принадлежностью к конкретному пользователю. Это будут следующие данные: ID результата, ID пользователя, название файла, путь к нему в файловой системе сервера, название процедуры и дату создания. Такая организация модели БД подразумевает использование СУБД с поддержкой SQL.

2.2 Выбор технологий разработки ПО

На этапе инициализации проекта в качестве основного языка программирования веб-приложения был выбран Python, это обоснованно тем, что в рассматриваемом раннее бизнес-процессе КАК-ЕСТЬ, на этапе визуализации и анализа – данные экспортировались в сторонние программы, где использовался функционал библиотек данного языка: matplotlib [19] и pandas [12]. Было решено не отступать от их использования, так как свои задачи они успешно выполняли, и использовать их при реализации функционала целевого веб-приложение. К тому же, к Python относятся фреймворки, которые часто используют для реализации подобных микросервисных веб-приложений [16]. Подбор дальнейших библиотек и стека технологий, а также обоснование их выбора – будет описано в таблице 5.

Таблица 5 – Выбор технологий и обоснование.

Название технологии/ библиотеки/ стороннего модуля	Описание	Обоснование выбора
Flask	Фреймворк для создания веб-приложений на Python	Flask был выбран из-за его легкости, гибкости и минималистичного подхода. В отличие от Django, Flask не навязывает жесткую структуру проекта, что позволяет адаптировать его под конкретные задачи. Это особенно важно для небольших проектов, где требуется быстрое прототипирование и минимальная избыточность. [13]
PostgreSQL	Реляционная система управления базами данных (СУБД)	Используется во многих проектах по разработке ПО в организации. Является очень надежной БД и поддерживает сложные запросы, а также полностью подходит под реализацию спроектированной модели БД. [19]
SQLAlchemy	Библиотека для работы с базами данных на Python (ORM)	Позволяет абстрагироваться от SQL-запросов [20] и работать с базой данных как с объектами Python. Это упрощает разработку и повышает читаемость кода. Прекрасно может работать с выбранной раннее PostgreSQL. [14]
Pg8000	Драйвер для подключения к PostgreSQL из Python	pg8000 был выбран как чистый Python-драйвер, который не требует дополнительных зависимостей и обеспечивает стабильное подключение к PostgreSQL
Flask-Login	Библиотека для управления аутентификацией пользователей в Flask.	Предоставляет простой и удобный способ управления сессиями пользователей, что критически важно для обеспечения безопасности и персонализации веб-приложения
Flask-WTF	Библиотека для работы с формами в Flask	Интегрируется с WTForms и упрощает создание и валидацию форм, что необходимо для регистрации, авторизации и загрузки данных.
Flask-SQLAlchemy	Интеграция SQLAlchemy с Flask для работы с базой данных	Упрощает настройку и использование SQLAlchemy в рамках Flask-приложения, что делает работу с базой данных более удобной
HTML/CSS/JS	Стандартные технологии для создания пользовательского интерфейса	Используются для создания интерактивного и отзывчивого интерфейса веб-приложения
WTForms	Библиотека для создания и валидации веб-форм на Python	Обеспечивает удобный способ работы с формами, включая валидацию данных, что важно для регистрации, авторизации и загрузки файлов.
Jinja2	Шаблонизатор для создания HTML-страниц в Flask.	Интегрирован с Flask и позволяет динамически генерировать HTML-страницы, что необходимо для отображения результатов анализа и визуализации.

Выбор технологического стека на таблице 5 выше строился исходя из определенных раннее требований по методологии FURPS+ в таблице, а также опыта предприятия и специфики решаемых задач.

По итогам выбора средств и технологий разработки этап проектирования был окончен, после чего был осуществлен переход к непосредственной реализации компонентов и функций.

2.3 Реализация функциональных требований

2.3.1. Настройка проекта

Первый шаг - реализация целевых программных модулей. В рамках его было создано виртуальное окружение для работы над проектом с помощью команды python -m venv venv, чтобы избежать конфликты зависимостей в последующем между компонентами. И вместе с этим была создана базовая структура проекта, которая показана на 12.

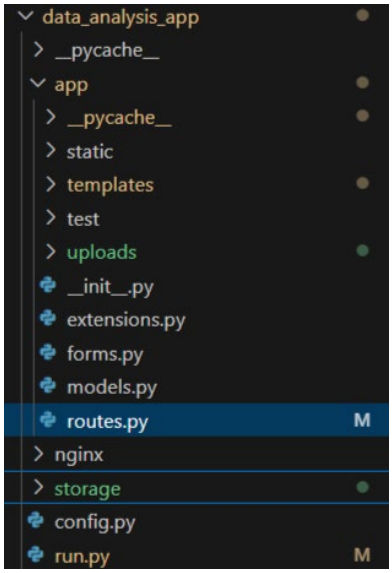


Рисунок 12 – Структура проекта

На данном рисунке показана общая структура проекта, но в рамках реализации поставленных задач имеют приоритетное значение следующие файлы и директории:

- Папка static – содержит в себе файлы с прописанными стилями, что подключаются к соответствующим страницам для реализации дизайна пользовательского интерфейса.
- Папка templates – содержит шаблоны HTML страниц, что в последующем будут динамически рендериться.
- __init__.py: стандартный файл для приложений на Python, служащий для инициализации модулей приложения;
- routes.py: основной файл проекта, содержащий выполняемые программой функции и машруты обработки HTTP запросов;
- models.py: файл, содержащий модели таблиц базы данных;
- extension.py: файл с некоторыми зависимостями проекта для избегания зацикленного импорта;
- forms.py: здесь прописаны шаблоны форм, что будут рендериться на HTML страницы с помощью Jinja2;
- config.py: хранит настройки конфигурации, которые используются в некоторых функциях инициализации проекта;
- run.py: файл запуска проекта, включающий способствующие отладки функции;

Описанная структура проекта обеспечивает модульность, четкое разделение логики и удобство дальнейшей разработки и поддержки приложения.

Листинги файлов будут приводиться далее в виде рисунком в соответствии с этапами реализации.

2.3.2 Реализация модулей регистрации и авторизации

В кодовом же виде она прописывается в раннее созданном файле models.py и выглядит следующим образом (рисунок 13):

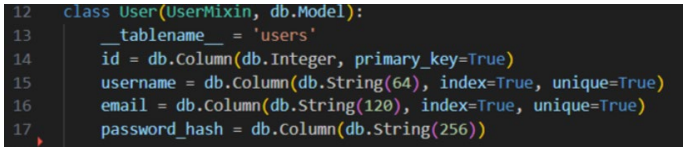


Рисунок 13 – Модель пользователя в коде программы

Здесь используется вспомогательный класс UserMixin, предоставляющий по умолчанию методы отслеживания текущего статуса пользователя, что будет использоваться в последующем.

Следующим шагом были созданы формы для регистрации и авторизации с использованием библиотеки Flask-WTF [8], предусматривающей валидацию конкретных полей форм (рисунок 14):

Рисунок 14 – Код моделей форм авторизации и регистрации

Форма авторизации предусматривает ввод пользователем его почты и пароля, а форма регистрации – имя пользователя, почту, пароля и его повторения.

Далее, были прописаны маршруты обработки форм (рисунок 15):

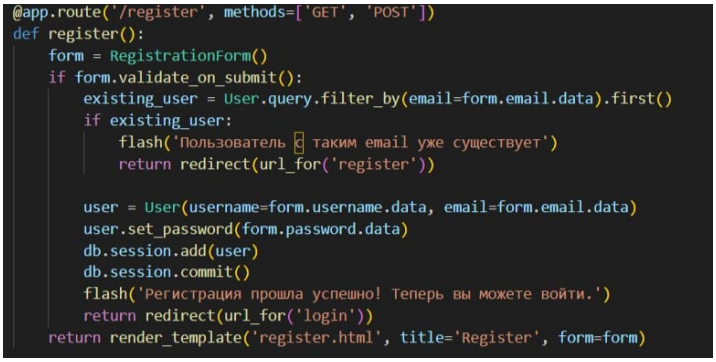


Рисунок 15 – Маршрут обработки формы регистрации

Данный маршрут по GET HTTP запросу с веб-приложения обрабатывает введенные пользователем данные в форме регистрации. Он производит запрос в БД и производит поиск пользователя по почте, так как это значение этого ключа является уникальными и предусматривают лишь единичное использование одной почты пре регистрации. Если на эту почту регистрация уже произведена, пользователя оставляет на этой же странице по POST запросу и выводит об этом сообщение.

Следующий маршрут обработки уже формы авторизации приведен на рисунке 16.

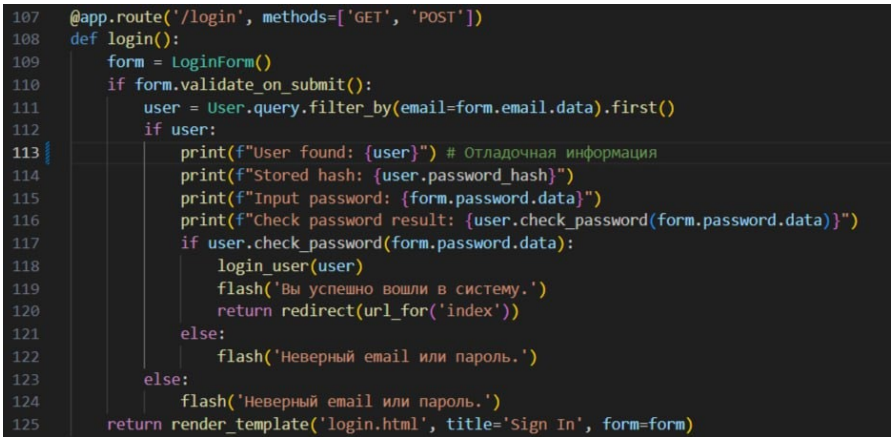


Рисунок 16 – Маршрут для авторизации

Маршрут обрабатывает введенные пользователем данные в форму авторизации и делает запрос в БД с поиском введенной почте. Если пользователь в базе отсутствует, то происходит редирект на эту же страницу с отображением сообщения пользователю о неправильно введенных данных. Если же по введенной почте существует пользователей, выполняется функция

check_password. Это стандартный метод Flask-Login и на вход принимает hash-значения и пароль, который так же переводит в хэш, и возвращает True или False в зависимости от того, совпадают ли они. Первое значение у нас подается в функции из БД, где сохраняются хэши паролей каждого пользователя, а пароль – из заполненного пользователем поля формы.

Последним шагов был проработан маршрут выхода пользователя из его аккаунта (рисунок 17):

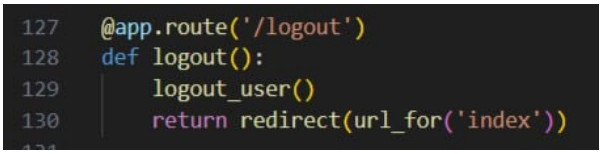


Рисунок 17 – Маршрут выхода пользователя из системы

Реализация функций маршрута происходит по нажатию пользователем кнопки выхода. Он использует функцию logout_user() – функцию вспомогательного класса UserMixin, упомянутого раннее, изменяющую статус пользователя на сайте.

2.3.3 Реализация модуля загрузки CSV-файла

В начале данного этапа необходимо было создать форму для загрузки файлов. Ее кодовая реализация модели приведена на рисунке 18.

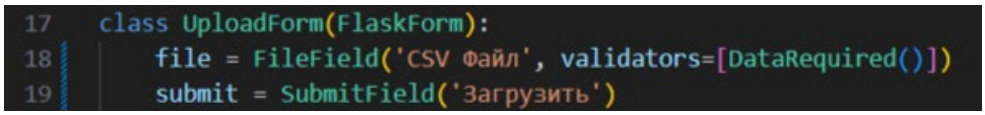


Рисунок 18 – Модель формы для загрузки файлов

Для обработки загруженных пользователем файлов необходимо прописать логику. По условиям задания необходимо было реализовать предпросмотр файла программой перед дальнейшими обработками. Предпросмотр представляет собой сбор общей статистики файла по определенным показателям и

отображение этой информации затем пользователю. Кодовая реализация маршрута с прописанной логикой приведена на рисунке 19.

```python
@app.route('/upload', methods=['GET', 'POST'])
@login_required
def upload():
    form = UploadForm()
    if form.validate_on_submit():
        file = form.file.data
        if file and file.filename.endswith('.csv'):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)

            try:
                file.save(filepath)
                result = check_csv_file(filepath)

                if result['is_valid']:
                    # Очищаем старое имя файла из сессии
                    session.pop('uploaded_file', None)
                    # Сохраняем новое имя файла в сессии
                    session['uploaded_file'] = filename
                    flash(result['message'])
                    return render_template('upload_result.html', result=result)
                else:
                    flash(result['message'])
                    return redirect(url_for('upload'))
            except Exception as e:
                print(f"Ошибка при обработке файла: {e}")
                flash('Ошибка при загрузке файла.')
        else:
            flash('Пожалуйста, загрузите файл в формате CSV.')
    return render_template('upload.html', form=form)
```

Рисунок 19 – Маршрут для обработки загружаемых файлов

Файл, загруженный пользователем, первым делом проходит проверку на формат, так как программа должна принимать только CSV-файлы, и если файл имеет иное расширение – происходит редирект на эту же страницу с отображением сообщения предупреждения. Если файл успешно проходит проверку, он сохраняется в файловой системе сервера по пути, заданному в файле конфигурации. После происходит предпросмотр файла с помощью функции check_csv_file, показанной на рисунке 20.

```python
def check_csv_file(filepath):
    try:
        df = pd.read_csv(filepath)

        if df.empty:
            return {
                'is_valid': False,
                'message': 'Файл пустой. Невозможно провести анализ.',
                'columns': [],
                'row_count': 0,
                'column_count': 0,
                'empty_cells': {},
                'total_cells': 0
            }
        row_count, column_count = df.shape
        empty_cells = df.isna().sum().to_dict()
        total_cells = row_count * column_count

        return {
            'is_valid': True,
            'message': 'Файл успешно загружен и проверен.',
            'columns': df.columns.tolist(),
            'row_count': row_count,
            'column_count': column_count,
            'empty_cells': empty_cells,
            'total_cells': total_cells
        }
    except Exception as e:
        return {
            'is_valid': False,
            'message': f'Ошибка при обработке файла: {e}',
            'columns': [],
            'row_count': 0,
            'column_count': 0,
            'empty_cells': {},
            'total_cells': 0
        }
```

Рисунок 20 – Рисунок функции предпросмотра файла

Данная функция проверяет файл, и собирает следующие данные, которые затем выводит пользователю в статистике на отдельной странице:

– Количество строк;

– Количество столбцов;

– Общее количество ячеек;

– Названия всех столбцов;

– Пустое количество ячеек по каждому столбцу.

Теперь, когда основные компоненты веб-приложения реализованы, был осуществлен переход к реализации целевого функционала – анализа данных и визуализации по пользовательским CSV-файлам.

2.3.4 Разработка механизмов анализа данных и визуализация.

На текущем этапе реализовывались основные функции приложения, определенные раннее. Так как на этапе проектирования веб-приложения было решено, что анализ и визуализация будет выполняться программой по заранее прописанным шаблонам, в которых пользователям будет задавать необходимые

ему параметры, если того предусматривает сам шаблон. Соответственно, для реализации этого необходимо каждый шаблон прописывать в коде как форму. Используя данный подход будет реализован принцип модульности, так как при необходимости добавления дополнительных шаблонов анализа или визуализации – будет намного проще их внедрять в новые версии приложения, а также расширять параметры настройки текущих. На рисунке 21 представлена реализация форм для проведения анализа данных.

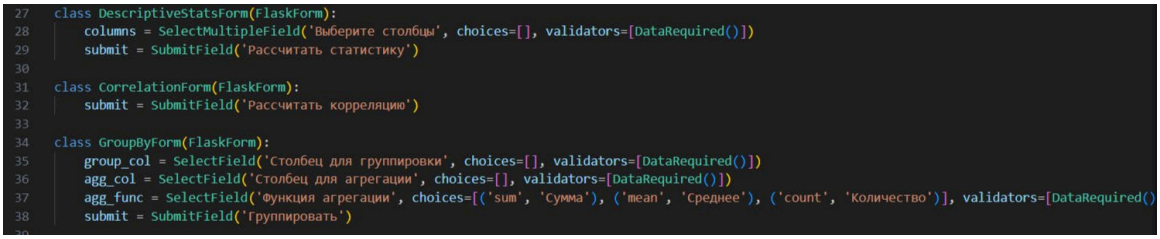
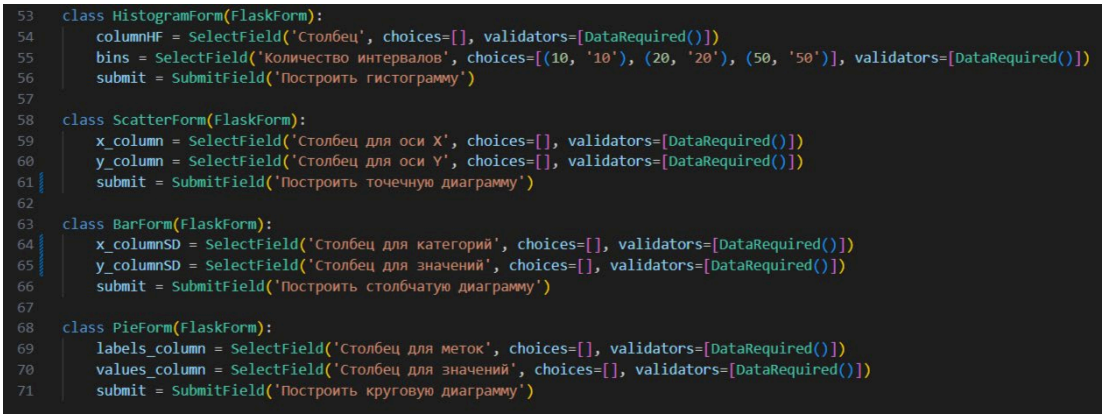


Рисунок 21 – Формы раздела анализа данных

В текущей реализации будет три вида анализа:

- статистический: Пользователю для проведения данного анализа необходимо будет выбрать в его форме столбцы, по которым программой будет приведена описательная статистика;
- корреляционный: Необходимости в его настройке для пользователя не будет, лишь только выбрать его самого в меню управления. Программа автоматически определит числовые столбцы в файле и сделает по ним корреляционный анализ;
- группировка: Пользователю будет предоставлено выбрать столбцы для группировки и агрегации, и функцию по которой она будет проводиться.

По такому же принципу реализованы были модели форм визуализации (рисунок 22):



```python
class HistogramForm(FlaskForm):
    columnHF = SelectField('Столбец', choices=[], validators=[DataRequired()])
    bins = SelectField('Количество интервалов', choices=[(10, '10'), (20, '20'), (50, '50')], validators=[DataRequired()])
    submit = SubmitField('Построить гистограмму')

class ScatterForm(FlaskForm):
    x_column = SelectField('Столбец для оси X', choices=[], validators=[DataRequired()])
    y_column = SelectField('Столбец для оси Y', choices=[], validators=[DataRequired()])
    submit = SubmitField('Построить точечную диаграмму')

class BarForm(FlaskForm):
    x_columnSD = SelectField('Столбец для категорий', choices=[], validators=[DataRequired()])
    y_columnSD = SelectField('Столбец для значений', choices=[], validators=[DataRequired()])
    submit = SubmitField('Построить столбчатую диаграмму')

class PieForm(FlaskForm):
    labels_column = SelectField('Столбец для меток', choices=[], validators=[DataRequired()])
    values_column = SelectField('Столбец для значений', choices=[], validators=[DataRequired()])
    submit = SubmitField('Построить круговую диаграмму')
```

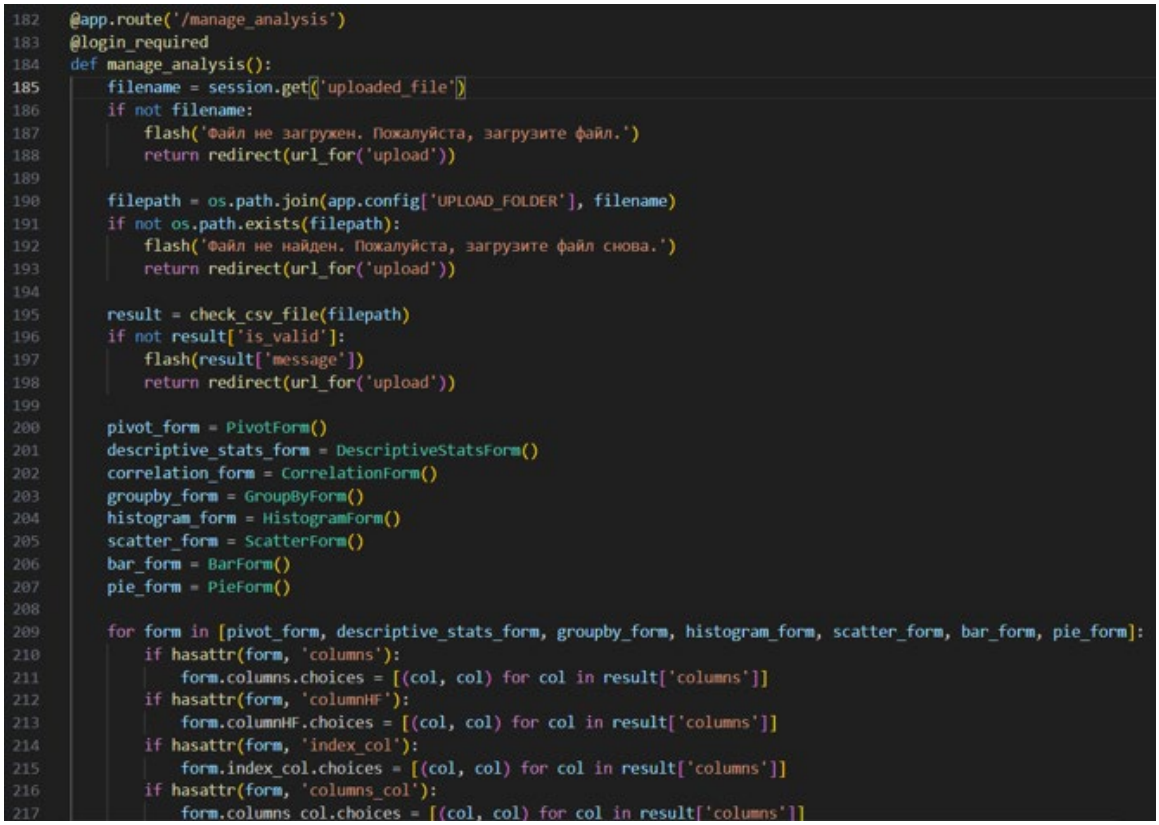
Рисунок 22 – Шаблоны раздела визуализации данных

В рамках реализации приложения, количество предлагаемых видов визуализации будут следующие:

- Гистограмма: Пользователь сможет выбрать столбец с данными и количество интервалов, на основе которых будет построена визуализация указанного типа;
- Точечная диаграмма: В содержании формы данного шаблона визуализации можно будет задать значения столбца для оси X и для оси Y;
- Столбчатая диаграмма: В ее настройках необходимо выбрать столбец для категорий и столбец со соответствующими ему значения, по которым будет выстроена данная диаграмма;
- Круговая диаграмма: Требует указания таких же параметров, как и для столбчатой диаграммы.

Список шаблонов визуализации и анализа является базовым, и предполагается что в процессе поддержке ПО будут добавляться новые шаблоны и настройки уже имеющихся в соответствии с нуждами организации. Такая реализация соответствует требованию модульности ПО, так как позволяет таким же путем создавать модели форм для новых типов.

Теперь, для работы с параметрами данных шаблонов и их выполнении по ним нужных функций, необходимо прописать им логику в файле маршрутов (рисунок 23):

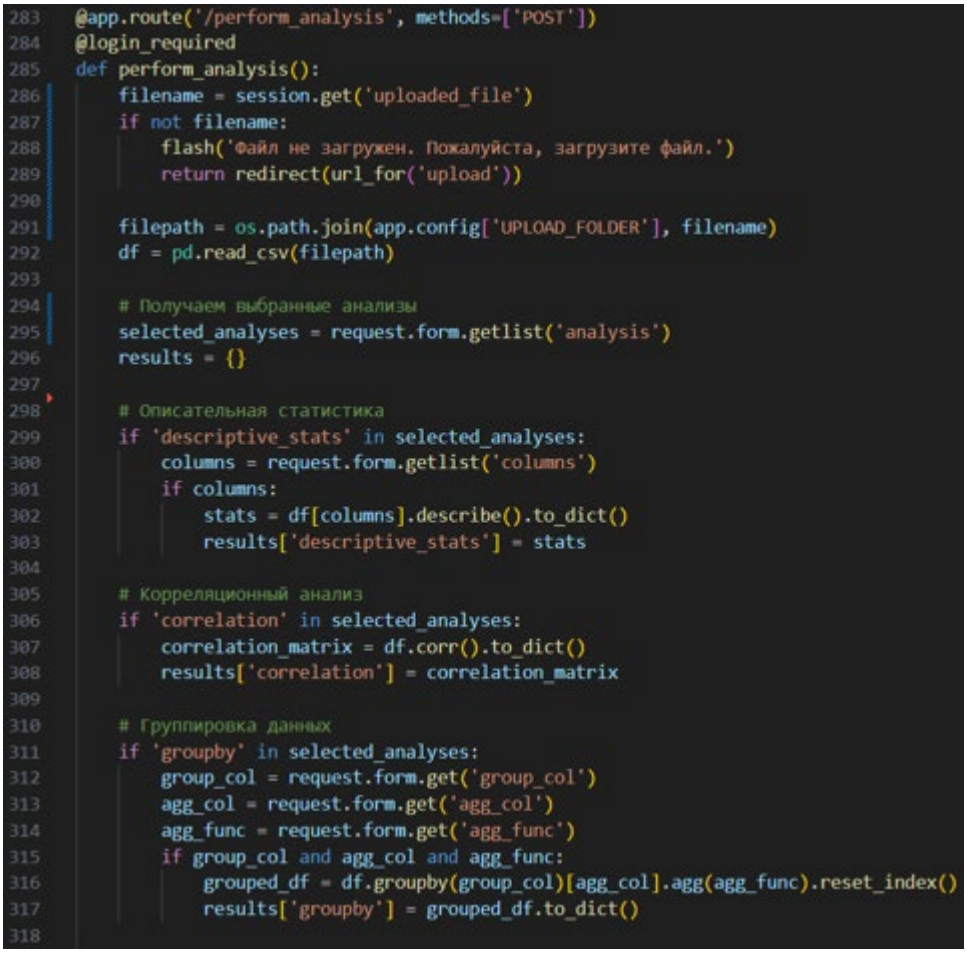


```python
@app.route('/manage_analysis')
@login_required
def manage_analysis():
    filename = session.get('uploaded_file')
    if not filename:
        flash('Файл не загружен. Пожалуйста, загрузите файл.')
        return redirect(url_for('upload'))

    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    if not os.path.exists(filepath):
        flash('Файл не найден. Пожалуйста, загрузите файл снова.')
        return redirect(url_for('upload'))

    result = check_csv_file(filepath)
    if not result['is_valid']:
        flash(result['message'])
        return redirect(url_for('upload'))

    pivot_form = PivotForm()
    descriptive_stats_form = DescriptiveStatsForm()
    correlation_form = CorrelationForm()
    groupby_form = GroupByForm()
    histogram_form = HistogramForm()
    scatter_form = ScatterForm()
    bar_form = BarForm()
    pie_form = PieForm()

    for form in [pivot_form, descriptive_stats_form, groupby_form, histogram_form, scatter_form, bar_form, pie_form]:
        if hasattr(form, 'columns'):
            form.columns.choices = [(col, col) for col in result['columns']]
        if hasattr(form, 'columnHF'):
            form.columnHF.choices = [(col, col) for col in result['columns']]
        if hasattr(form, 'index_col'):
            form.index_col.choices = [(col, col) for col in result['columns']]
        if hasattr(form, 'columns_col'):
            form.columns_col.choices = [(col, col) for col in result['columns']]
```

Рисунок 23 – Маршрут рендеринга форм анализа и визуализации

Функция данного маршрута дополнительно проходит по содержанию загруженного файла и собирает из него данные, которые будут заполняться в списки параметров шаблонов. При успешном проходе, функция переадресует пользователя на страницу с шаблонами и создает на ней формы, заполняя их списки доступными к выбору параметрами.

Следующим шагом была прописана логика выполнения операций по выбранным шаблонам и на рисунке 24 приведена кодовая реализация логики выполнении операций анализа.



```python
@app.route('/perform_analysis', methods=['POST'])
@login_required
def perform_analysis():
    filename = session.get('uploaded_file')
    if not filename:
        flash('Файл не загружен. Пожалуйста, загрузите файл.')
        return redirect(url_for('upload'))

    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    df = pd.read_csv(filepath)

    # Получаем выбранные анализы
    selected_analyses = request.form.getlist('analysis')
    results = {}

    # Описательная статистика
    if 'descriptive_stats' in selected_analyses:
        columns = request.form.getlist('columns')
        if columns:
            stats = df[columns].describe().to_dict()
            results['descriptive_stats'] = stats

    # Корреляционный анализ
    if 'correlation' in selected_analyses:
        correlation_matrix = df.corr().to_dict()
        results['correlation'] = correlation_matrix

    # Группировка данных
    if 'groupby' in selected_analyses:
        group_col = request.form.get('group_col')
        agg_col = request.form.get('agg_col')
        agg_func = request.form.get('agg_func')
        if group_col and agg_col and agg_func:
            grouped_df = df.groupby(group_col)[agg_col].agg(agg_func).reset_index()
            results['groupby'] = grouped_df.to_dict()

```

Рисунок 24 – Логика выполнения анализов по шаблонам

В данном маршруте прописана функция, которая берет из сессии название файла, загруженного пользователем и получает по запросу список выбранных пользователем анализов и их параметров, и далее по прописанным для каждого анализа алгоритму анализа – выполняет его. Выполнение происходит посредством исходных методов pandas [15]. Результат сохраняется в сессии и результаты анализов сохраняются в HTML файл, а сам файл – сохраняется в файловой системе сервера с указанием в названии даты выполнения анализа. Далее пользователя переводит на страницу просмотра результата анализов, а метаданные файла результатов сохранятся в БД с помощью функции save_result_to_db. Функция сохраняет данные как экземпляр модели результата и приведена на рисунке 25.

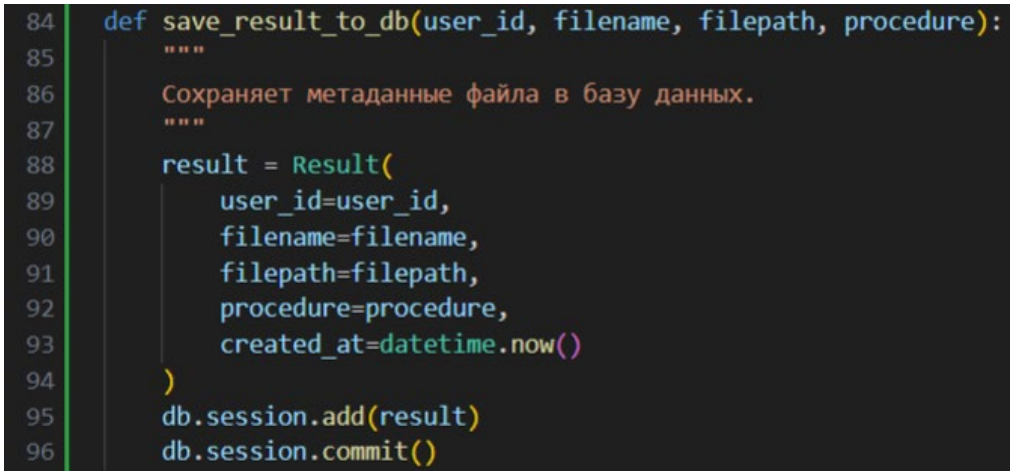


Рисунок 25 – Листинг функции save_result_to_db

Она принимает по созданию результата по шаблону значения переменных и присваивает их в соответствии с моделью результата, приведенной на рисунке 26, после чего вносит их в таблицу базы данных PostgreSQL.

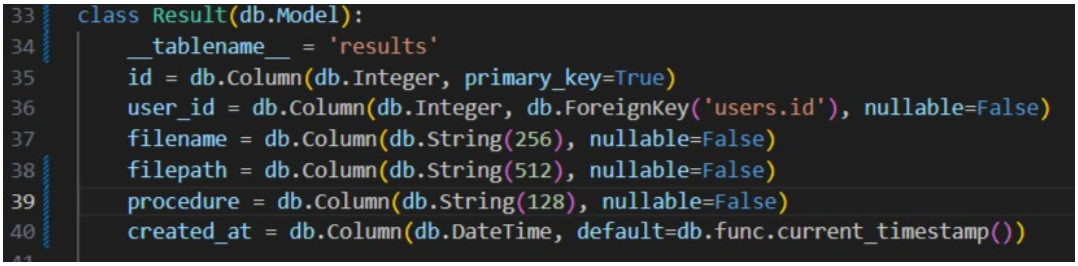
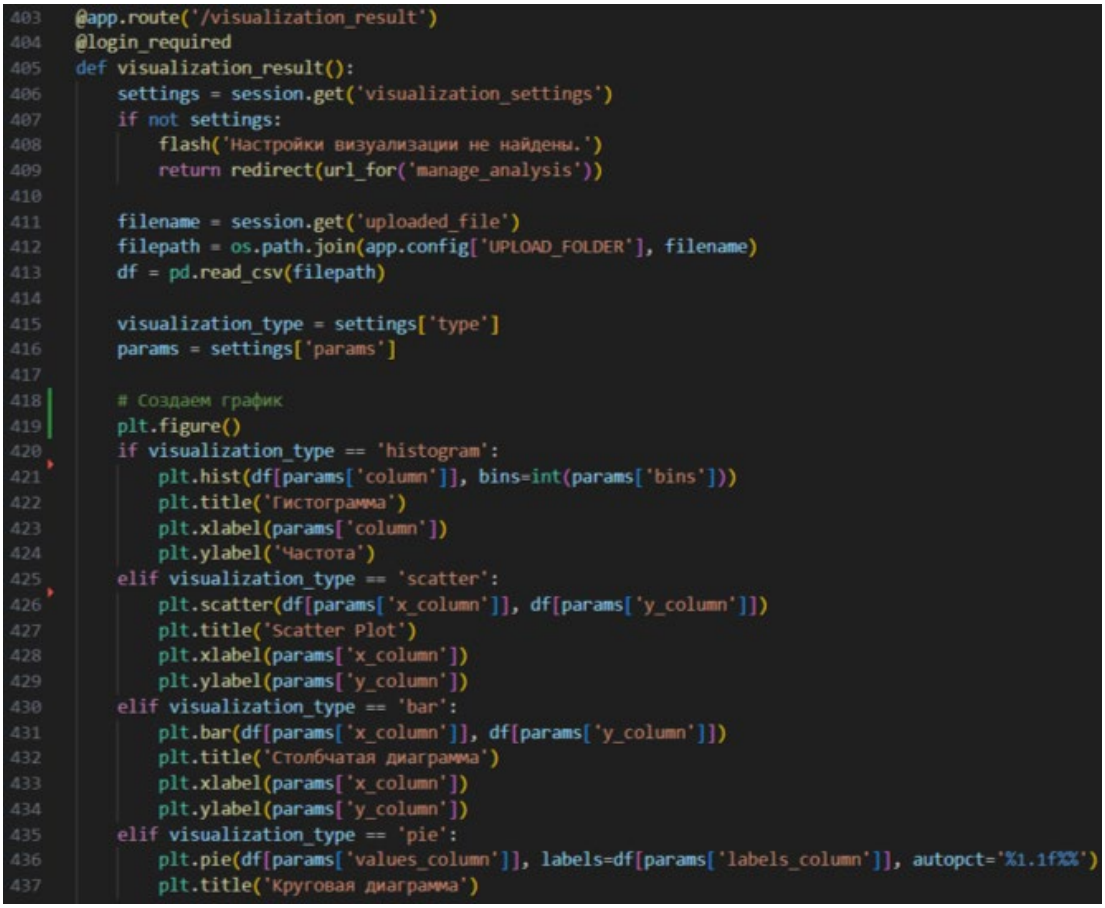


Рисунок 26 – Модель результата

Функция принимает на вход ID пользователя, название файла, путь к нему в файловой системе сервера и названия выполненных процедур, после составляет экземпляр результата и вносит его в базу данных.

Теперь необходимо было реализовать логику для выполнения визуализации и сохранения ее результатов. Ее листинг приведен на рисунке 27.



```python
@app.route('/visualization_result')
@login_required
def visualization_result():
    settings = session.get('visualization_settings')
    if not settings:
        flash('Настройки визуализации не найдены.')
        return redirect(url_for('manage_analysis'))

    filename = session.get('uploaded_file')
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    df = pd.read_csv(filepath)

    visualization_type = settings['type']
    params = settings['params']

    # Создаем график
    plt.figure()
    if visualization_type == 'histogram':
        plt.hist(df[params['column']], bins=int(params['bins']))
        plt.title('Гистограмма')
        plt.xlabel(params['column'])
        plt.ylabel('Частота')
    elif visualization_type == 'scatter':
        plt.scatter(df[params['x_column']], df[params['y_column']])
        plt.title('Scatter Plot')
        plt.xlabel(params['x_column'])
        plt.ylabel(params['y_column'])
    elif visualization_type == 'bar':
        plt.bar(df[params['x_column']], df[params['y_column']])
        plt.title('Столбчатая диаграмма')
        plt.xlabel(params['x_column'])
        plt.ylabel(params['y_column'])
    elif visualization_type == 'pie':
        plt.pie(df[params['values_column']], labels=df[params['labels_column']], autopct='%1.1f%%')
        plt.title('Круговая диаграмма')
```

Рисунок 27 – Логика выполнения визуализации и сохранения графиков

Функция маршрута, как и в случае с анализом, берет из сессии данные о загруженном файле и выбранных параметрах визуализации, и, исходя из выбранного типа визуализации, выполняет его по алгоритму с помощью библиотеки matplotlib и ее методов [19]. С помощью функции save_result_to_db сохраняет файл в метаданные файла в БД, после чего переадресовывает пользователя на страницу просмотра результатов и рендерит там результат визуализации.

На каждой страницы просмотра результатов предусмотрена кнопка скачивания результатов, после нажатия которой выполняются соответствующие типу результата функции.

2.3.5 Обеспечение безопасности веб-приложения

Так как основное предназначение приложения состоит в предоставлении пользователям инструментария для анализа и визуализации их данных, из этого

следует факт того, что пользователи могут загружать «чувствительные» данные, можно сделать вывод, что необходимо в реализации применить меры защиты пользовательских данных.

В реализации до этого уже было предпринято несколько мер, таких как:

- SQLAlchemy: Используется для отправки параметризованных запросов в базу данных PostgreSQL, исключая тем самым возможность выполнения злоумышленниками SQL-инъекций для получения пользовательских данных из БД или входа под чужим логином;
- FlaskWTF: Помимо валидации полей форм по определенным типам, еще и экранирует вводимые пользователем данные, что также пресекает исполнение SQL-инъекций;
- Хранение данных в БД и bcrypt: Устроено так, что при сохранении пользовательских данных для авторизации – сам пароль не записывается в БД, но записывается его hash-ключ, который создается при помощи библиотеки bcrypt. Тем самым, если злоумышленник каким-то образом сможет получить доступ к БД, он не сможет получить пользовательские пароли.

Но помимо вышеописанных мер, необходимо применить в реализации еще несколько для дополнительного обеспечения безопасности пользовательских данных.

В связи с этим было принято решение использовать Flask-Talisman. Он является внутренним модулем фреймворка и добавляет заголовок Content-Security-Policy, именуемый далее, как CSP, с безопасными настройками по умолчанию, однако использование его стандартной конфигурации не покрывает всего спектра угроз, в следствии чего пришлось дополнять политику CSP дополнительными директивами [2]. Прописанная политика CSP в коде показана на рисунке 28.

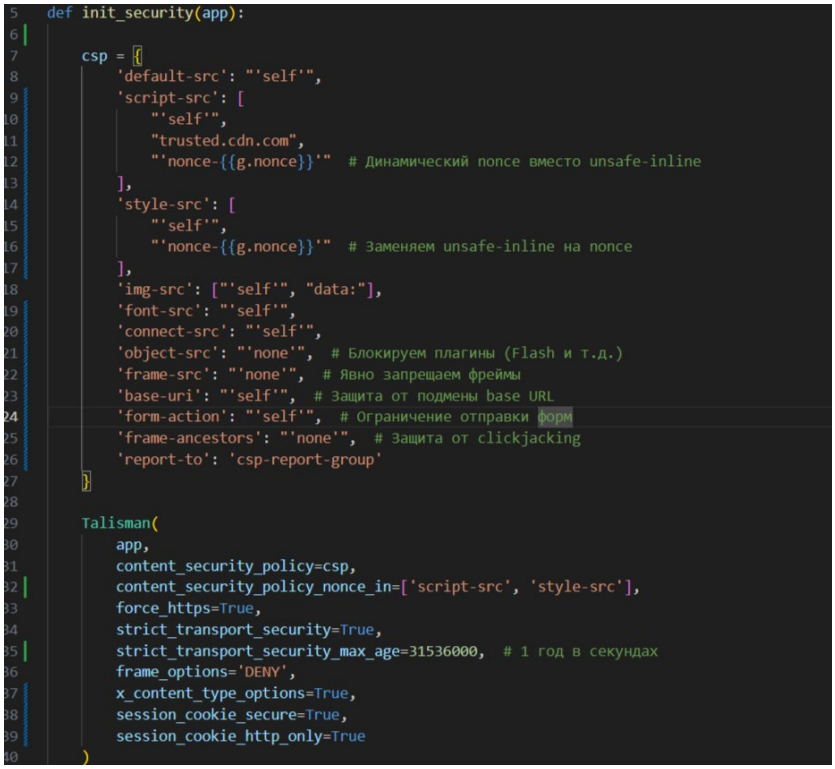


Рисунок 28 – Политика CSP приложения

Директивы, связанные с устранением выявленных уязвимостей в коде помечены напротив комментариями с пояснением своей необходимости.

Следующие изменения касались файла конфигурации веб-приложения, и были они связаны с уязвимостью в виде cookie без атрибута. Частично здесь были предприняты меры еще в настройке конфигурации CSP, но дополнительно было необходимо добавить атрибут Lax для cookie-файлов, как это показано на рисунке 29.

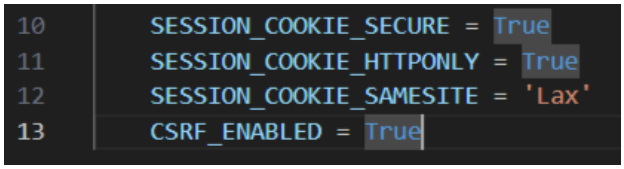


Рисунок 29 – Добавление атрибута Lax для cookie-файлов

Теперь, когда атрибут добавлен и CSRF включен, угроза возможности отправления злоумышленниками файлов куки со сторонних сайтов полностью пресечена.

Последующие действия касаются предотвращения утечки информации о версии сервера. Так как на данный момент приложения находится еще

Реализовано, она запускается на сервере для отладки и разработки самого Flask – Werkzeug, и данное явление типичное для ситуации и легко закрывается, когда программа выходит на продакшен. Но в целях избавления от всех уязвимостей, предпримем меры по закрытию данной утечки.

Для покрытия уязвимости утечки – необходимо запустить веб-приложение через WSGI и в его параметрах задать запрет на показ информации о версиях. Для этого был выбран и предварительно установлен Nginx. Перед запуском, открываем его файл конфигурации и редактируем его, как показано на рисунке 30.

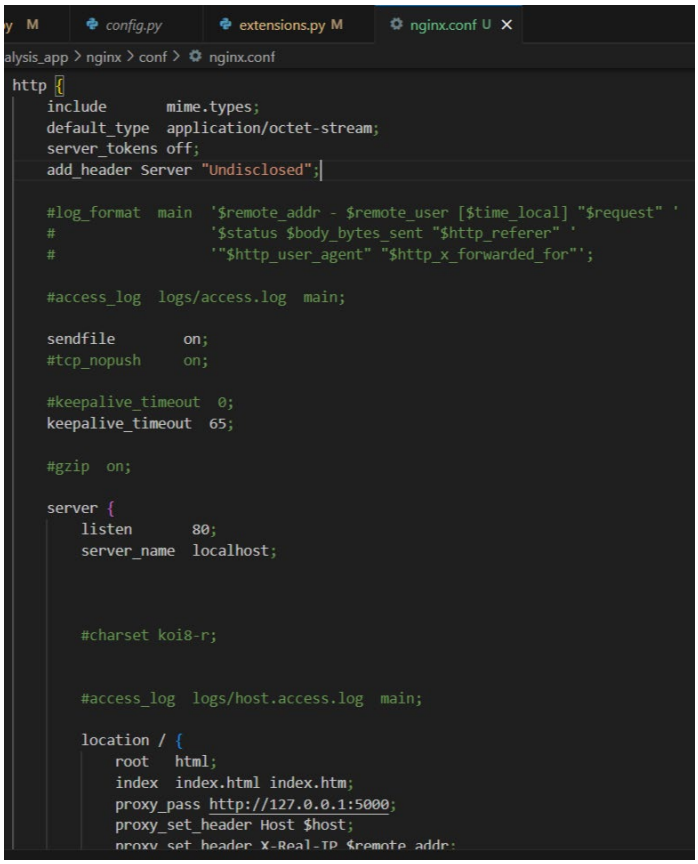


Рисунок 30 – Конфигурация Nginx

А именно, указываем чтобы в заголовке вместо любой информации о сервере, была лишь строка «Undisclosed» и в location добавляем в прокси ссылку на веб-приложение.

После того, как все меры по устранению угроз были успешно предприняты, перейдем к реализации дополнительных мер безопасности, связанных с базой данных.

Одной из проблем являлось, что сервер БД запускает на порте 5432, который выставлен по умолчанию известен всем и является первым, что проверяют злоумышленники. Для его смены было необходимо перейти в файл конфигурации (рисунок 31):

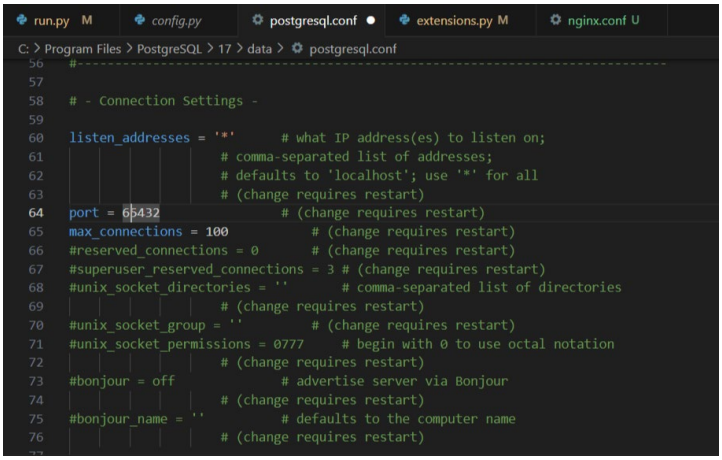


Рисунок 31 – Содержимое файла конфигурации PostgreSQL

Значением порта было успешно изменено с 5432 на 65432, с таким и сохраняем конфигурацию.

Дополнительно организуем шифрование все трафика между клиентом и сервером, защищая от перехвата данных. Предварительно скачав библиотеки OpenSSL, добавляем ее в .PATH. Был создан каталог для сертификатов и в командной строке была введена команда для генерации, где указываем путь к папке для них (рисунок 32):

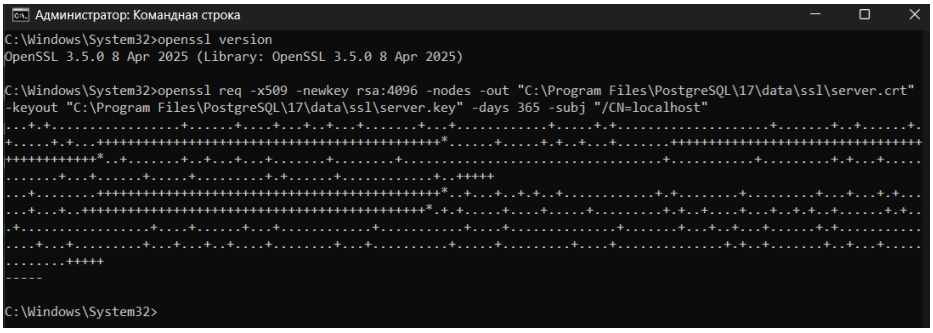


Рисунок 32 – Генерация SSL сертификатов

На рисунке 32 показан процесс успешного выполнения генерации сертификатов, после чего дополнительно было проверено в созданной раннее директории их наличие (рисунок 33):

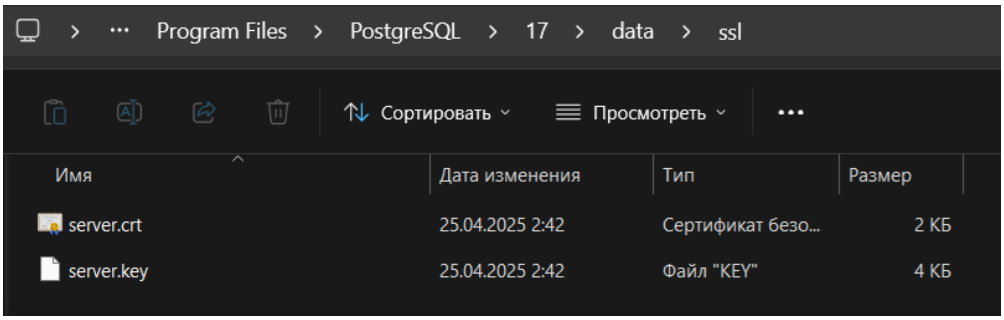


Рисунок 33 – Директория со сгенерированным SSL сертификатами

Как можно заметить на рисунке 33, сертификат и ключ были успешно сгенерированы и находятся в нужной папке.

– server.crt - публичный сертификат;
– server.key - приватный ключ.

Далее, отправляемся снова в файл конфигурации и включаем в нем SSL и указываем путь к сертификату и ключу, как показано на рисунке 34.

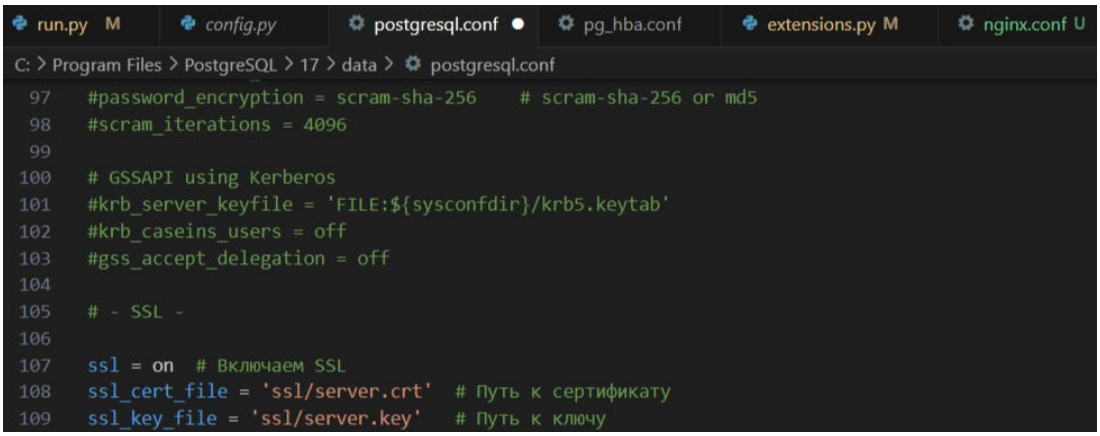


Рисунок 34 – Конфигурация PostgreSQL

После изменения конфигурации база данных была успешно перезапущена с вступленными в силу изменениями.

Последним шагом была реализация бэкапов БД, чтобы в случае каких-либо происшествий с базой данных – потери были минимизированы, и пользователи не теряли все свои данные.

Для этого создаем скрипт, которые будет делать дампы базы данных в специальный каталог для бэкапов и удалять бэкапы старше 30 дней. Скрипт показан на рисунке 35.

```
$date = Get-Date -Format "yyyy-MM-dd"
$backupPath = "C:\PostgreSQL\backups\$date"

# Создаем каталог для бэкапа
mkdir $backupPath -Force

# Делаем дамп базы
& "C:\Program Files\PostgreSQL\17\bin\pg_dump.exe" -U postgres -h localhost -p 65432 -F c -b -f "$backupPath\mydb.dump" mydb

# Удаляем бэкапы старше 30 дней
Get-ChildItem "C:\PostgreSQL\backups" | Where-Object {$_.CreationTime -lt (Get-Date).AddDays(-30)} | Remove-Item -Recurse -Force
```

Рисунок 35 – Скрипт для бэкапов экземпляров БД

Данный скрипт был успешно выполнен в PowerShell и дамп БД успешно создается в соответствующей директории. Результаты работы скрипта показаны на рисунке 36.

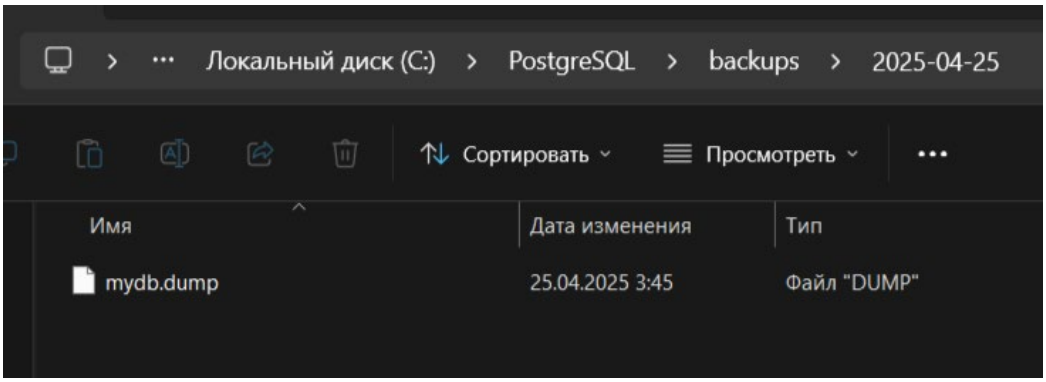


Рисунок 36 – Результат выполнения скрипта создания дампа

При внедрении можно использовать на сервере расположения БД данный скрипт и задать ему периодичность, по которой он будет выполняться и создавать бэкапы базы данных и удалять старые.

После применения Реализовано веб-приложения описанных выше мер, можно считать данный этап завершенным и переходить к следующему.

2.3.6 Реализация пользовательского интерфейса

Последний этап выполнения поставленных задач на реализацию определенных требований к приложению. В ходе данного этапа было необходимо реализовать интерфейс, то есть проработать HTML шаблоны страниц для динамической генерации и файлы стилей. Все шаблоны были реализованы отталкиваясь от их описаний в таблице на этапе проектирования.

Первым делом, необходимо было создать базовый шаблон, который в зависимости от действий пользователя в веб-приложении, будет расширяться другими шаблонами. Его HTML-код представлен на рисунке 37.

Рисунок 37 – HTML-код базового шаблона

В базовом шаблоне, представленном на рисунке 44. присутствуют фрагменты Jinja2 для связывания пользовательского интерфейса с серверной логикой приложения. Он предусматривает отображение навигационного меню.

Далее, реализация шаблонов для главного меню, страницы регистрации и авторизации. Шаблон главной страницы приведен на рисунке 38.

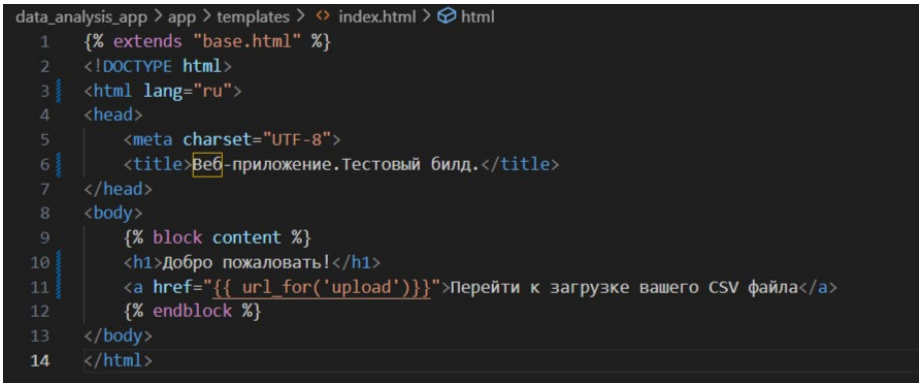


Рисунок 38 – Шаблон главной страницы

На главной странице реализованы элемент перехода на страницу загрузки файла и несколько контейнер для наполнения страницы, а также специальной кнопкой, видимой только неавторизованным пользователям с предложением выполнить авторизацию для пользования функционалом. Следующим был реализован шаблон страницы авторизации и его код приведен на рисунке 39.

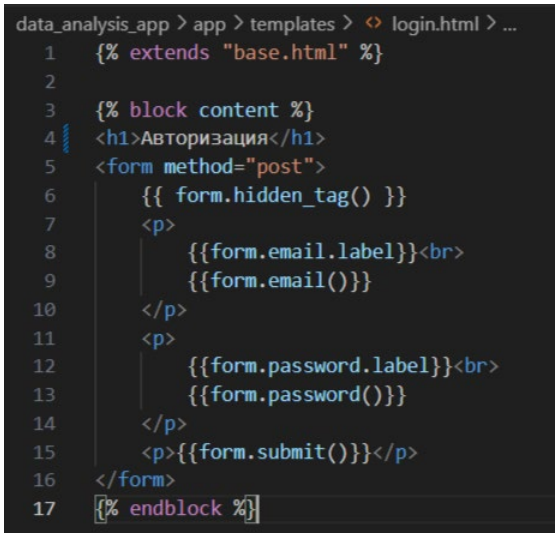


Рисунок 39 – Шаблон страницы авторизации

На шаблоне страницы авторизаций реализованы поля, в соответствии с ее моделью формы, созданной раннее – то есть полем ввода логина, пароля и кнопкой подтверждения. И далее был реализован шаблон страницы регистрации (рисунок 40):

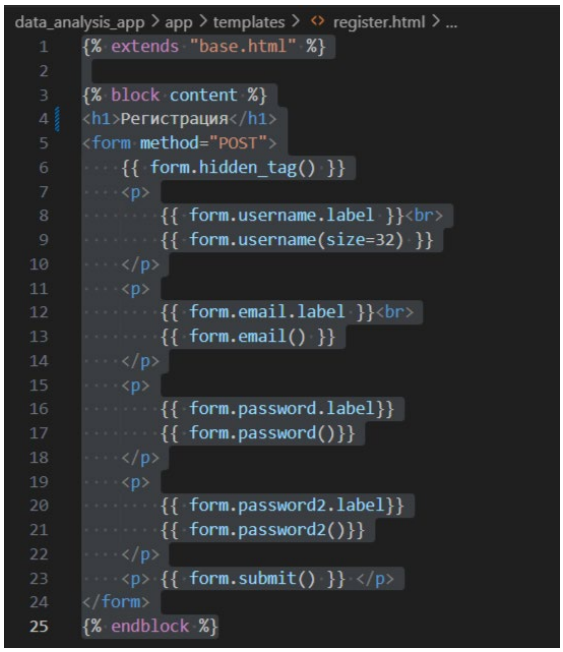


Рисунок 40 – Шаблон страницы регистрации

В следующем шаге, необходимо реализовать шаблоны страниц для загрузки файлов и страницы предпросмотра файла. На первой необходимо отобразить форму загрузки по уже написанному в приложении шаблону, а во второй – отобразить наглядно и удобно для пользователя собранные предпросмотром данные. Реализованный HTML-код страницы загрузки файла показан на рисунке 41.

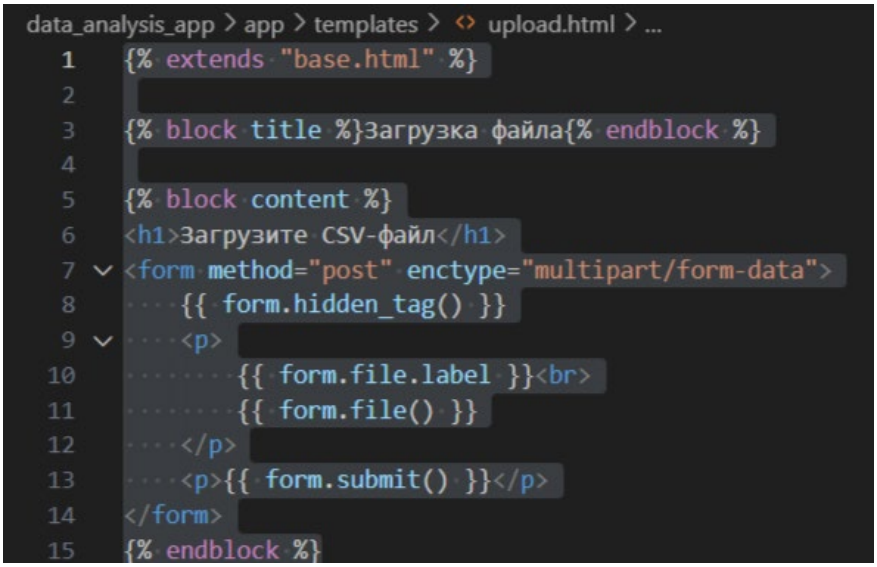


Рисунок 41 – Шаблон страницы загрузки файла

В шаблоне страницы загрузки файла, показанном на рисунке 41. реализовано поле, при нажатии на которую пользователю можно будет выбрать файл для выгрузки на своем устройстве и кнопка подтверждения выбора.

Последним реализованным шаблоном веб-страницы была страницы предпросмотра файла и ее листинг представлен на рисунке 42.

data_analysis_app > app > templates > upload_result.html > ...

```
{% extends "base.html" %}

{% block extra_css %}
    <link rel="stylesheet" href="{{ url_for('static', filename='upload_result.css') }}">
{% endblock %}

{% block content %}
<h1>Результаты проверки файла</h1>

{% if result.is_valid %}
    <p>{{ result.message }}</p>
    <h2>Информация о файле:</h2>
    <ul>
        <li><strong>Количество строк:</strong> {{ result.row_count }}</li>
        <li><strong>Количество столбцов:</strong> {{ result.column_count }}</li>
        <li><strong>Общее количество ячеек:</strong> {{ result.total_cells }}</li>
    </ul>

    <h2>Столбцы:</h2>
    <ul>
        {% for column in result.columns %}
            <li>{{ column }}</li>
        {% endfor %}
    </ul>

    <h2>Пустые ячейки по столбцам:</h2>
    <ul>
        {% for column, count in result.empty_cells.items() %}
            <li><strong>{{ column }}:</strong> {{ count }} пустых ячеек</li>
        {% endfor %}
    </ul>

    <a href="{{ url_for('manage_analysis') }}">Перейти к управлению анализами</a>
{% else %}
    <p>{{ result.message }}</p>
    <a href="{{ url_for('upload') }}">Попробовать снова</a>
{% endif %}
```

Рисунок 42 – Шаблон страницы предпросмотра файла

На данной странице будет отображаться информация о файле, то есть количество строк, столбцов и ячеек, а также опции для дальнейших действий при выявлении в загруженном файле пустых строк.

В завершении данного этапа идет написание набора многочисленных файлов статики (CSS, JS), что будут задавать дизайн написанным выше шаблонам и организовывать их функционал на стороне клиента. Цвет проекта – фиолетовый градиент, он будет задавать фон и необходимо подстроить все элементы под него, чтобы все удобно отображалось для пользователя приложения и отчетливо было видно.

Выводы по главе 2.

Во второй главе выпускной квалификационной работы на основе требований, сформулированных в первой главе по методологии FURPS, была спроектирована архитектура программного обеспечения, включающая в себя диаграмму вариантов использования, компонентов, развертывания. Также была разработана физическая модель базы данных для хранения информации о пользователях и результатах их работ. Был осуществлен выбор стека технологий реализации проекта с обоснованием каждого решения. В рамках реализации веб-приложения, были разработаны ключевые модули системы: регистрация и авторизация пользователей, загрузка CSV-файлов, выполнения анализа и визуализации данных по шаблонам, обеспечение безопасности. На последнем этапе был разработан пользовательский интерфейс с использованием HTML, CSS, JavaScript и шаблонизатора Jinja2.

3 Оценка эффективности разработанного программного обеспечения

3.1 Тестирование функционала приложения

По окончанию написания кода веб-приложения, обязательно протестировать приложение. Тестирование является важным этапом разработки и нуждается в комплексном проведении.

Для его проведения использовалась утилита pytest. Предварительно были написаны unit-тесты, направленные на проверку функций приложения и работы различных связанных модулей друг с другом. Тесты были прописаны в отдельном файле test.py в директории проекта и в таблице будут подробно описаны.

Таблица 6 - Таблица тестовых сценариев

ID сценария	Название сценария	Шаги тестирования	Ожидаемый результат	Фактический результат	Статус
AS001	Вход в систему	1. Открыть страницу входа 2. Ввести правильный логин и пароль 3. Нажать кнопку "Войти"	Пользователь успешно вошел	Пользователь успешно вошел	Пройден
AS002	Неправильный вход	1. Открыть страницу входа 2. Ввести неправильный логин и/или пароль 3. Нажать кнопку "Войти"	Отобразить сообщение об ошибке	Отобразить сообщение об ошибке	Пройден
AS003	Регистрация нового пользователя	1. Открыть страницу регистрации 2. Заполнить все поля 3. Нажать кнопку "Зарегистрироваться"	Пользователь зарегистрирован	Пользователь зарегистрирован	Пройден

Продолжение таблицы 6

ID сценария	Название сценария	Шаги тестирования	Ожидаемый результат	Фактический результат	Статус
FS001	Загрузить CSV-файл	1. Открыть страницу загрузки файла 2. Выбрать файл формата CSV в файловой системе клиента 3. Нажать кнопку “Upload”	Пользователь успешно загрузил файл на сервер и был переведен на страницу предпросмотра	Пользователь успешно загрузил файл на сервер и был переведен на страницу предпросмотра	Пройден
FS002	Загрузка файла иного формата, кроме CSV	1. Открыть страницу загрузки файла 2. Выбрать файл формата любого формата, кроме CSV в файловой системе клиента 3. Нажать кнопку “Upload”	Отобразить сообщение об ошибке	Отобразить сообщение об ошибке	Пройден
ANS001	Проведение анализа данных по одному шаблону	1. Загрузить CSV-файл 2. Выбрать шаблон “Описательная статистика” 3. Нажать кнопку “Провести анализ”	Пользователь был перенесен на страницу просмотра результатов. Результаты соответствуют выбранному типу анализа.	Пользователь был перенесен на страницу просмотра результатов. Результаты соответствуют выбранному типу анализа.	Пройден
VS001	Проведение визуализации по шаблону	1. Загрузить CSV-файл 2. Выбрать шаблон “Круговая диаграмма” 3. В настройке шаблона задать столбцы с названиями и данными. 4. Нажать кнопку “Произвести визуализацию”	Пользователь был перенесен на страницу просмотра результатов. Результаты соответствуют выбранному типу визуализации.	Пользователь был перенесен на страницу просмотра результатов. Результаты соответствуют выбранному типу визуализации.	Пройден
FS003	Скачивание результатов анализа	1. На странице просмотра результатов анализа нажать кнопку “Скачать”	Результаты анализа скачиваются	При попытке скачать файл в менеджере загрузок браузера показывается сообщение “Ошибка сервера”	Ошибка

Продолжение таблицы 6

FS004	Скачивание результатов визуализации	1. На странице просмотра результатов анализа нажать кнопку “Скачать”	Результаты визуализации скачиваются	Результаты визуализации скачиваются	Пройден
AS003	Переход в личный кабинет	1. На главной странице нажать кнопку “Личный кабинет”	Личный кабинет успешно открывается и пользователю предоставляются данные о последних результатах	Личный кабинет успешно открывается и пользователю предоставляются данные о последних результатах	Пройден
FS005	Скачивание результатов из личного кабинета	1. На странице личного кабинета нажать кнопку “Скачать” рядом с результатом анализа. 2. На странице личного кабинета нажать кнопку “Скачать” рядом с результатом визуализации	Результаты анализа и визуализации успешно скачиваются	Результаты анализа и визуализации успешно скачиваются	Пройден
AS004	Выход из аккаунта	1. На главной странице нажать кнопку “Выйти”	Производится успешный выход из учетной записи	Производится успешный выход из учетной записи	Пройден

В столбце шаги тестирования прописан алгоритм для проверки работы функций путем воссоздания сценария вручную.

Тестирование показало высокое качество основной функциональности приложения, однако, имеется недостаток, требующий внимания.

В отдельную таблицу 7 были вынесены сценарии тестирования, чей фактический результат отличается от ожидаемого.

Таблица 7 - Таблица отслеживания ошибок

ID ошибки	Описание ошибки	Шаги воспроизведения	Статус
BUG001	Ошибка при попытке скачать результаты анализа со страницы просмотра результатов	1. На странице просмотра результатов анализа нажать кнопку “Скачать”	Исправлено

После устранения ошибок было проведение регрессионного тестирования для проверки того, не повлияло исправление на работу других функций модуля или модулей связанных. По итогам его проведения все тесты, включая раннее проваленный, были успешно пройдены, тем самым можно сказать, что реализованные функции приложения работают так, как надо.

3.2 Тестирование безопасности

В данном разделе будет тестироваться безопасность приложения, меры по обеспечению которой были применены в процессе разработки.

Первоначальное ее тестирования предполагает использование SAST анализатора. В качестве статистического анализатора кода, поддерживающего Python, был выбран и предварительно установлен посредством pip install – Bandit.

Для выполнения анализа необходимо прописать в терминале bandit -r, после чего указать директорию проекта со всем ее кодом. Анализатор в таком случае проанализирует весь код и сделает лог с результатами по найденным уязвимостям. Результаты его работы показаны на рисунке 43

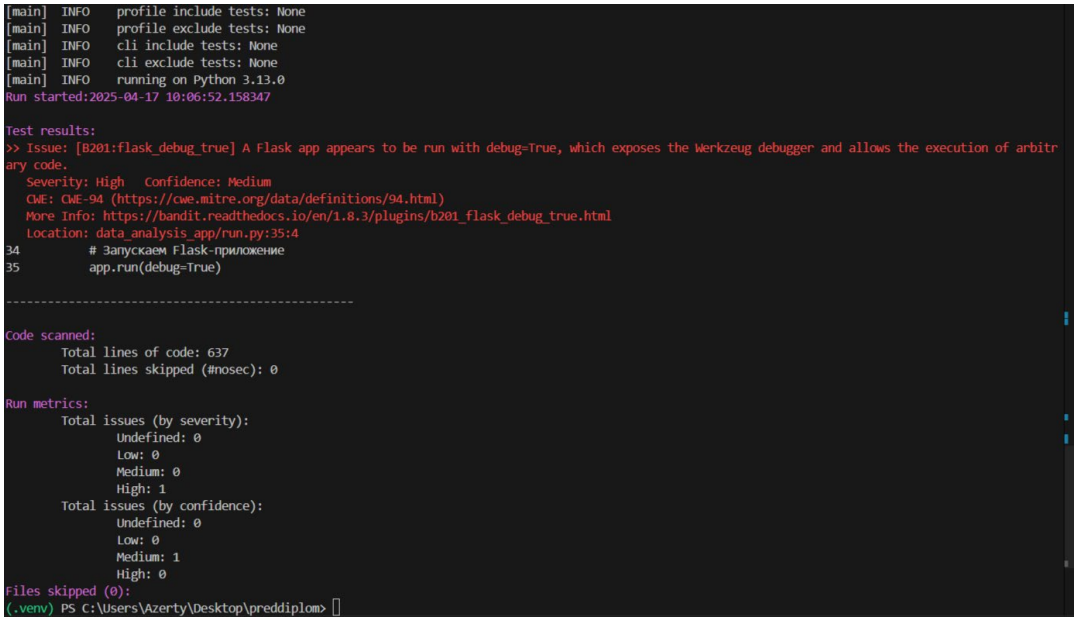


Рисунок 43 – Результаты работы статического анализатора

По итогам работы статического анализатора, была найдена лишь одна проблема, связанная с тем, что приложение запускается в debug-режиме, что естественно для всего этапа разработки веб-приложения до внедрения. При внедрении, разумеется, данный режим будет отключен посредством выставления в параметре запуска debug=False.

Следующим этапом тестирования предполагалось использования уже DAST-анализа. Они предназначены для атаки приложения в рамках тестирования его уязвимостей различными способами. И в качестве инструмента тут был выбран ZAP. Работает по принципу того, что принимает ссылку на веб-сайт приложения и проводит по данному адресу полноценную атаку [6]. Для атаки была выбрана форма авторизации приложения. Результаты его работы показаны на рисунке 44.

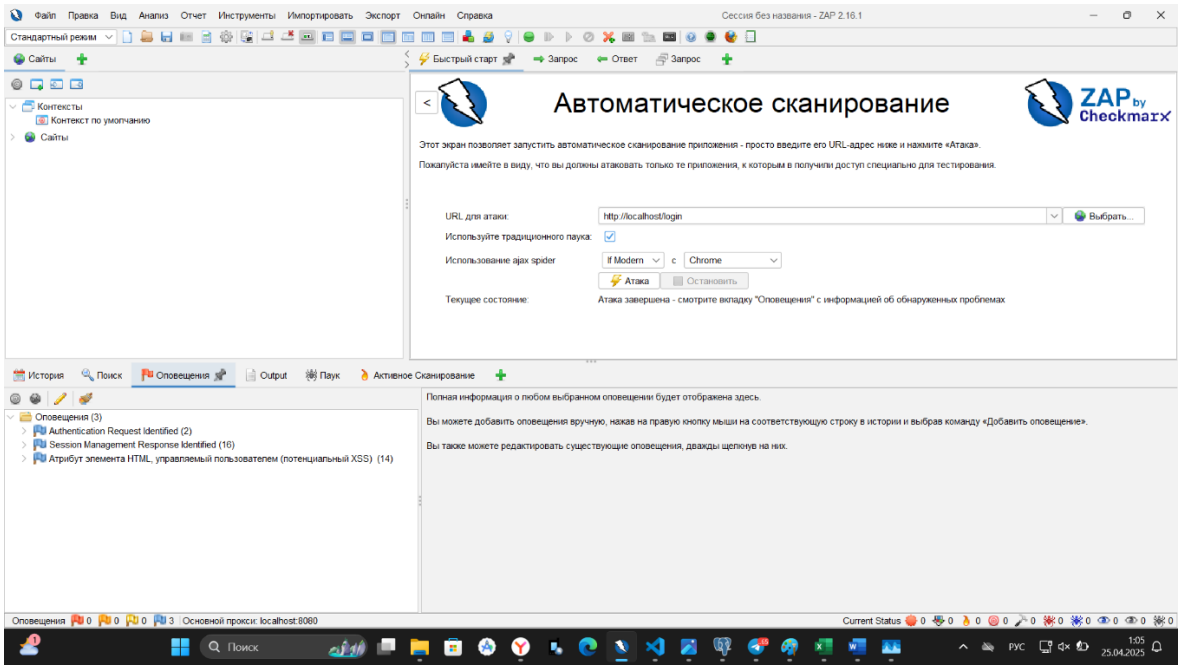


Рисунок 44 – Итоги работы DAST-анализатора ZAP

По итогам анализа, как и ожидалось, критических уязвимостей не было обнаружены в силу того, что во Flask-Talisman была прописана CSP и присвоен атрибут к cookie-файлам. Дополнительно стоит уточнить, что перед проверкой веб-приложение было запущено через Nginx, что так же предотвратило утечку версии сервера.

3.3 Контрольный пример и демонстрация.

После того, как все тесты успешно пройдены и баги исправлены, проводится контрольный пример, в ходе которого будет зафиксирован весь пользовательский путь от перехода на сайт, до создания результатов анализа и визуализации. В нем будут представлены все интерфейсы и наглядно продемонстрирована работа всех модулей разработанного веб приложения.

Начало было взято с момента, как незарегистрированный пользователь попадает на главную страницу веб-приложения (рисунок 45):

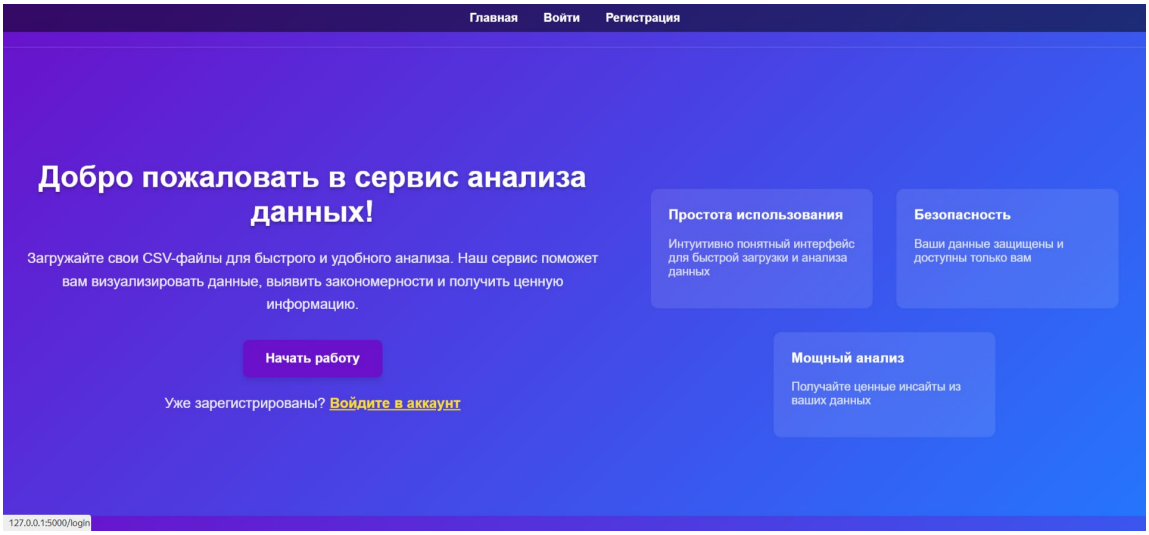


Рисунок 45 – Главная страница с экрана незарегистрированного пользователя

В соответствии с реализацией шаблона страницы, незарегистрированному пользователю отображается сообщение с вопрос о регистрации и ссылкой рядом с ней к форме авторизации. Так же навигационное меню сверху уникально для незарегистрированного пользователя.

Далее была нажата кнопка «Начать работу», но так как пользователь на авторизован, был выполнен редирект на страницу с формой регистрации, что является ожидаемым результатом. Сама страница регистрации показана на рисунке 46.

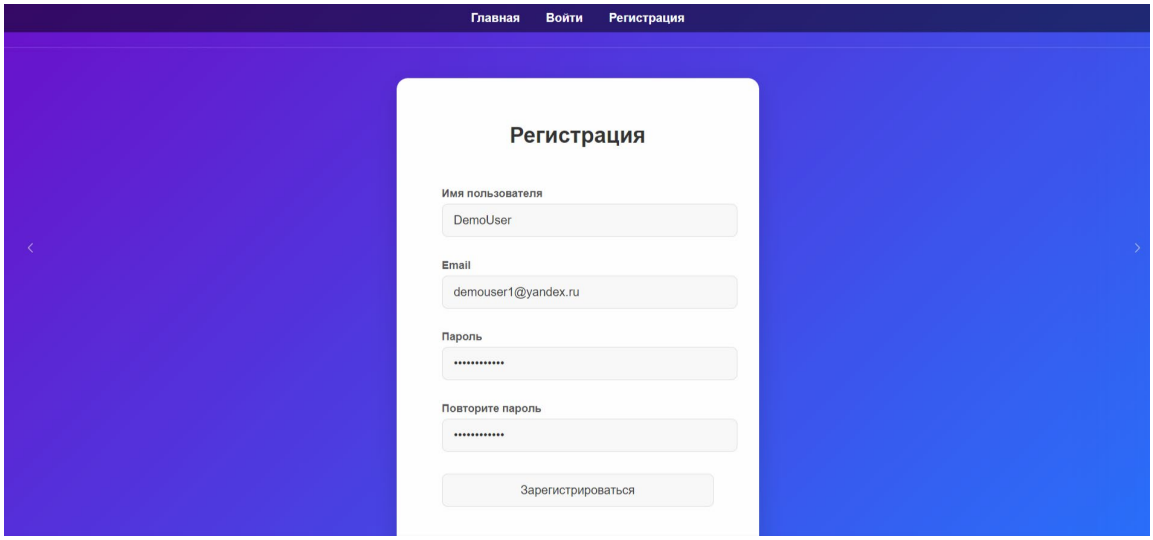


Рисунок 46 – Страница регистрации

В форме регистрации для дальнейшей демонстрации вводим необходимые данные для создания нового пользователя:

– Имя пользователя: DemoUser;
– Email: demouser1@yandex.ru;
– Пароль: DemoUser1872;
– Повторный пароль: DemoUser1872.

После чего, была нажата кнопка «Зарегистрироваться». Система успешно зарегистрирует пользователя, переадресовав его на страницу авторизации, с уведомлением, что теперь он с помощью своих данных может выполнить вход. На серверной стороне же данные будут обработаны и занесены в БД. Форма авторизации, на которую переадресовало пользователя, показана на рисунке 47.

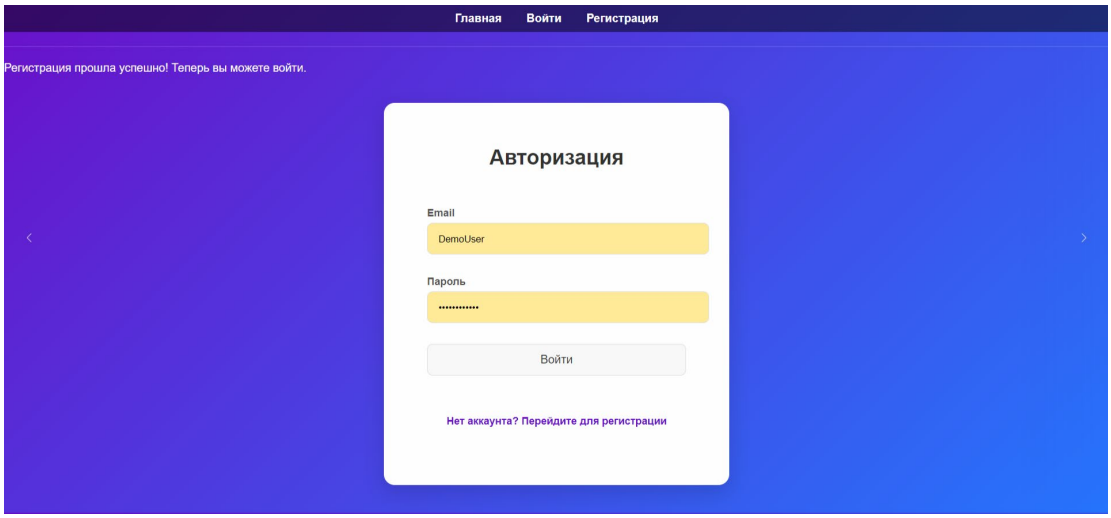


Рисунок 47 – Форма авторизации

Как можно увидеть на рисунке 47, уведомление об успешной регистрации отобразилось, что значит система занесла данные о пользователе в БД (рисунок 48):

	id [PK] integer	username character varying (64)	email character varying (120)	password_hash character varying (256)
4	7	Tester9999	tester9999@gmail.com	scrypt:32768:8:1$Djwur1HzLpcQE0PG$1028c6551f6937654c6cdd0d07d1a0e5b4d5b5a9ffc3f8a8d325e38983c832c5d01496c6f2cce06
5	1	Oleg	sanfra@gmail.com	$2a$06$2FsYn.CfBlI2SW4PLEjTROF7b.2VW8k6xAcaABMxlu3YBYP7hZPbu
6	3	Anastasiya	neil.bull@yandex.ru	scrypt:32768:8:1$ZUzEfViA5qNHxaH3$dda2911a5240f862bfa23486b4383d2713edac024d82713b1d9a487bcddbb512ef4d7240c2aa2e
7	4	Tester1111	sanyhthha@yahoo.com	scrypt:32768:8:1$Rp5srwChHRKrpAuF$d331d383348a6da87bf8f6b0bc2df2a9e63baf344df5214169f2d986db83a9f0cd45633795916d9
8	8	ZAP	zaproxy@example.com	scrypt:32768:8:1$4TYVqDw7YltPrxoO$4331da7b18baca134f8a273306d0f9ede89eb6598c1fcd72072add33c2c8f90908c8a42a617dc30
9	9	TestUser11	testuser11@mail.ru	scrypt:32768:8:1$uBCU2TAgCQijsEB0$2a6383ac21ad3c053103151c577767693963ed9fd523ed959938b6a0ae90d8f10d222f5e7d370a
10	10	DemoUser	demouser1@yandex.ru	scrypt:32768:8:1$Fxe4jLVt1DnXn2e2$4571ab475f8da6da1395628e33dd546c7041ba419d7171a36c2915ca05c32b386a90c20a785354

Рисунок 48 – Сохранение пользователя в БД

На рисунке также показано подтверждение факта регистрации, введенные пользовательские данные в соответствии с ключами были занесены в таблицу users.

Был произведен ввод данных, указанных в форме регистрации, в соответствующие поля уже формы авторизации и была нажата кнопка «Войти». На рисунке 49 можно увидеть уведомление, подтверждающее вход в систему, от чего можно сделать вывод, что модуль авторизации приложения работает корректно.

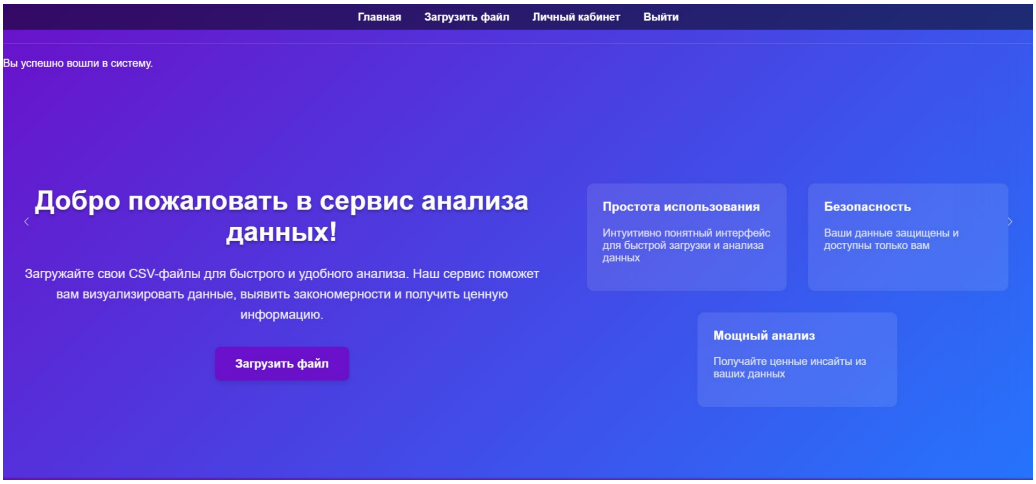


Рисунок 49 – Главная страница с роли авторизованного пользователя

Так же на рисунке 49 можно заметить, что для авторизованного пользователя страница выглядит иначе: доступно больше вкладок в навигационном меню и нету сообщения с вопросом об авторизации.

Идем далее по пути пользователя и начинаем работу с приложением. Начинается она с того, что пользователю необходимо загрузить в систему свой CSV-файл, для пользования основным функционалом веб-приложения. Для этого на главной странице приложения была нажата кнопка и осуществлен переход на страницу загрузки файла, что показана на рисунке 50.

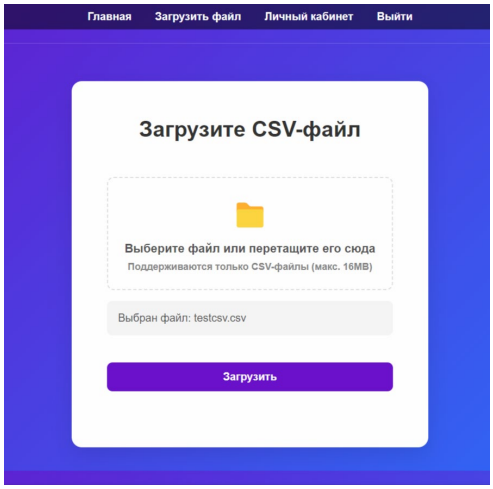


Рисунок 50 – Страница загрузки CSV-файла

Так она выглядит с уже выбранным пользователем файлом. Работает форму следующим образом - когда пользователь нажимает на иконку – открывается проводник, в котором он может выбрать желаемый для загрузки файл, если он соответствует требованию. Также форме присвоен атрибут drag-and-drop, благодаря которому можно загрузить файл прямым перетаскиванием его с устройства.

Загруженный файл, данные которого будут использоваться далее примерах реализации анализа и визуализации, имеет следующее содержание, показанное на рисунке 51.

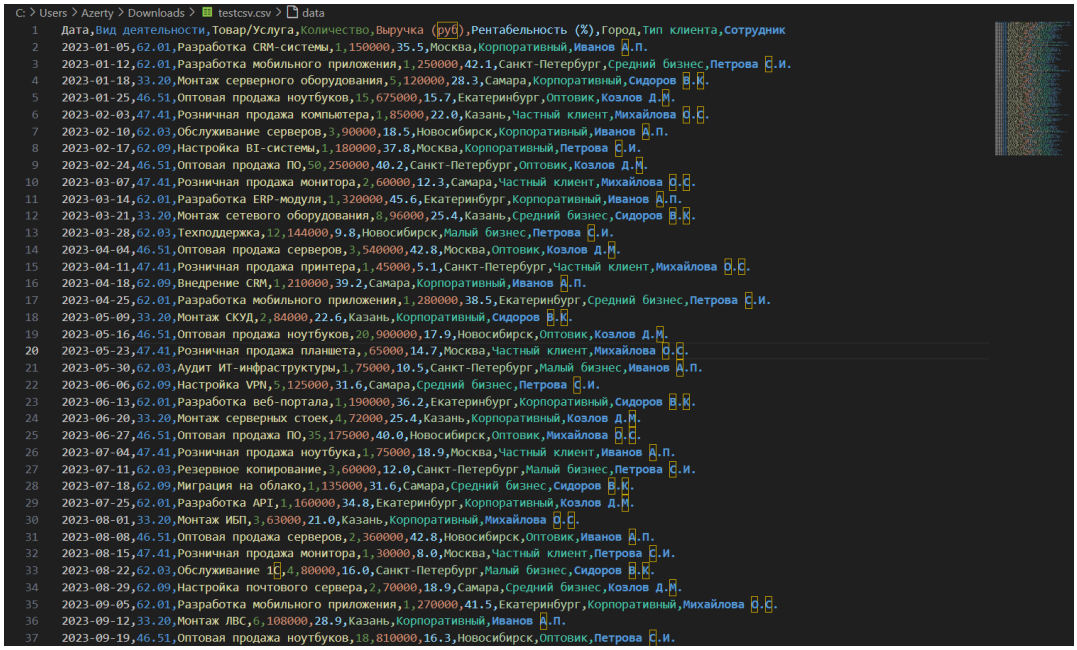


Рисунок 51 – Содержимое тестового файла CSV

Содержимое файла, показанного на рисунке, является типовым для файлов данных, находящихся в обороте организации ООО «Ворлдинтертех Рус». Так же, в содержимом файле, одна ячейка пуста, для того, чтобы проверить полностью модуль обработки CSV файла. На странице загрузки нажимается кнопка «Загрузить», после чего перебрасывается пользователь на страницу предпросмотра, показанную на рисунке 52.

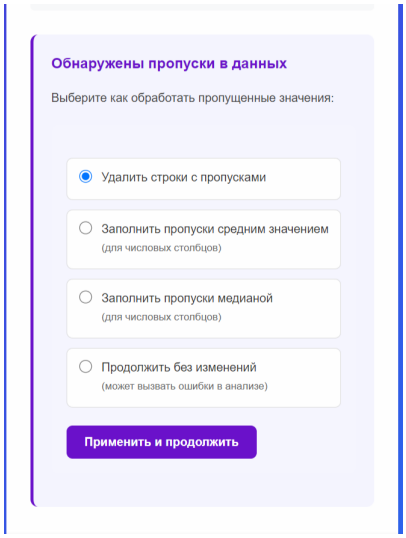


Рисунок 52 – Форма выбора обработки пропущенных значений в файле на странице предпросмотра

Форма предоставляет пользователю четыре опции и дополнительно показывает, для каких данных какая актуальна и какие могут быть последствия в случае некоторых. В рамках контрольного примера, так как пропущенного значение находится не в числовом столбце – была выбрана опция удаления строки для корректности анализа. После чего, была нажата кнопка «Применить и продолжить» с последующей переадресацией на страницу управления. Она показана на рисунке 53.

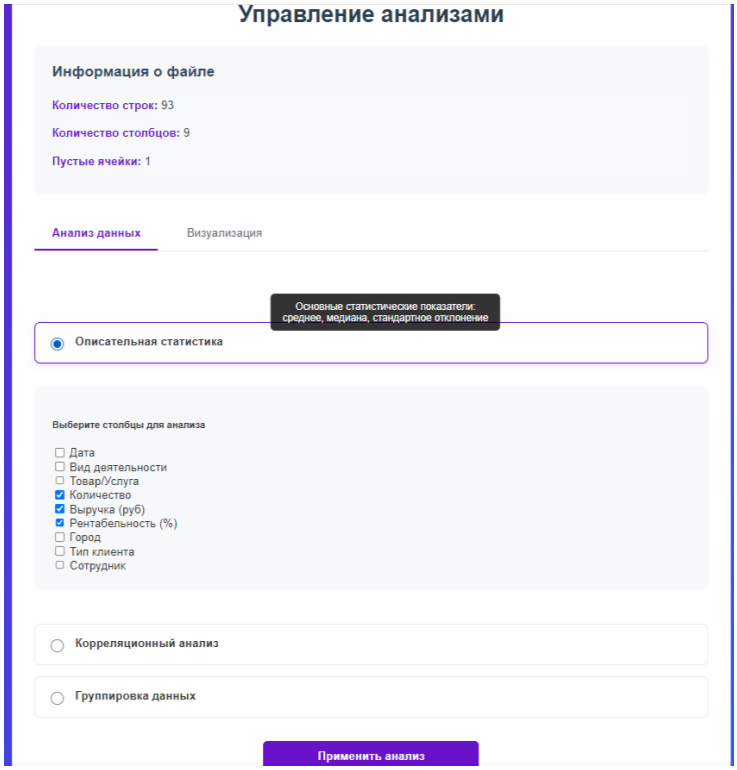


Рисунок 53 – Страница управления анализами

На данной странице представлены реализованные шаблоны анализов. При их выборе открываются параметры для их настройки, соответствующие конкретно этому шаблону. Так же при наведении на шаблон – высвечивается короткое пояснение по нему и его функции. Был выбран шаблон описательной статистики и для данного вида анализа были выбраны числовые столбцы загруженного раннее файла, а именно – Количество, Выручка, Рентабельность.

После того, как параметры были заданы, нажимаем «Применить анализ» и происходит перенаправление на страницу просмотра результатов выбранного анализа. Результаты его показаны на рисунке 54.

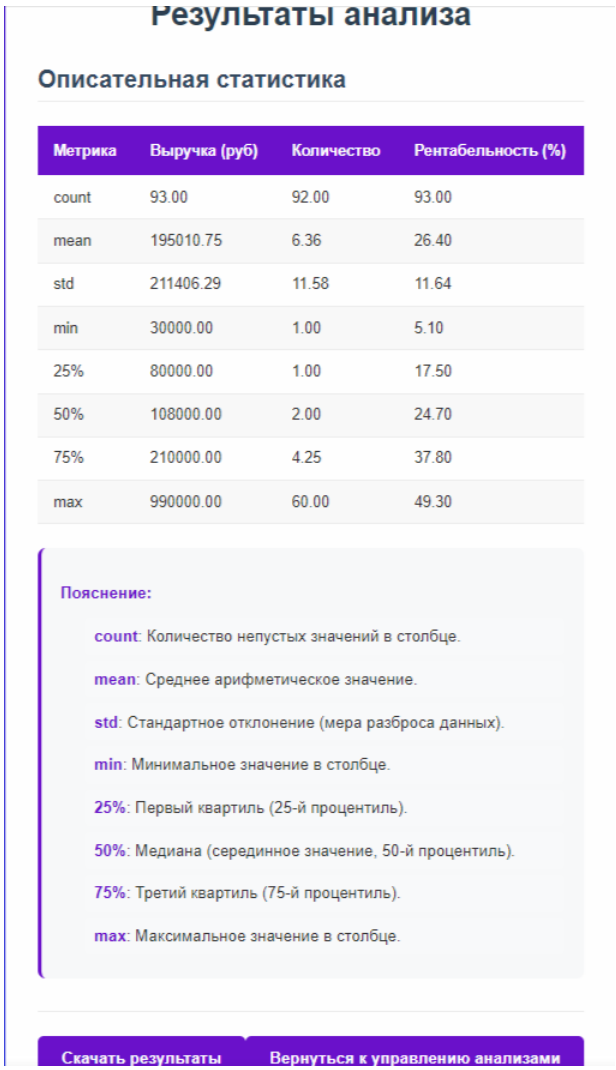


Рисунок 54 – Содержимое загружаемого CSV-файла

Как можно увидеть по рисунку 54, результаты в соответствии с заданными параметрами успешно выстроились и отобразились пользователю, и дополнительно к ним пояснение к метрикам описательной статистики, чтобы пользователю было проще понять.

Далее, был выполнен переход на страницу управления назад уже в меню визуализаций для демонстрации работы модуля визуализации. Страница управления с разделом визуализации показана на рисунке 55.

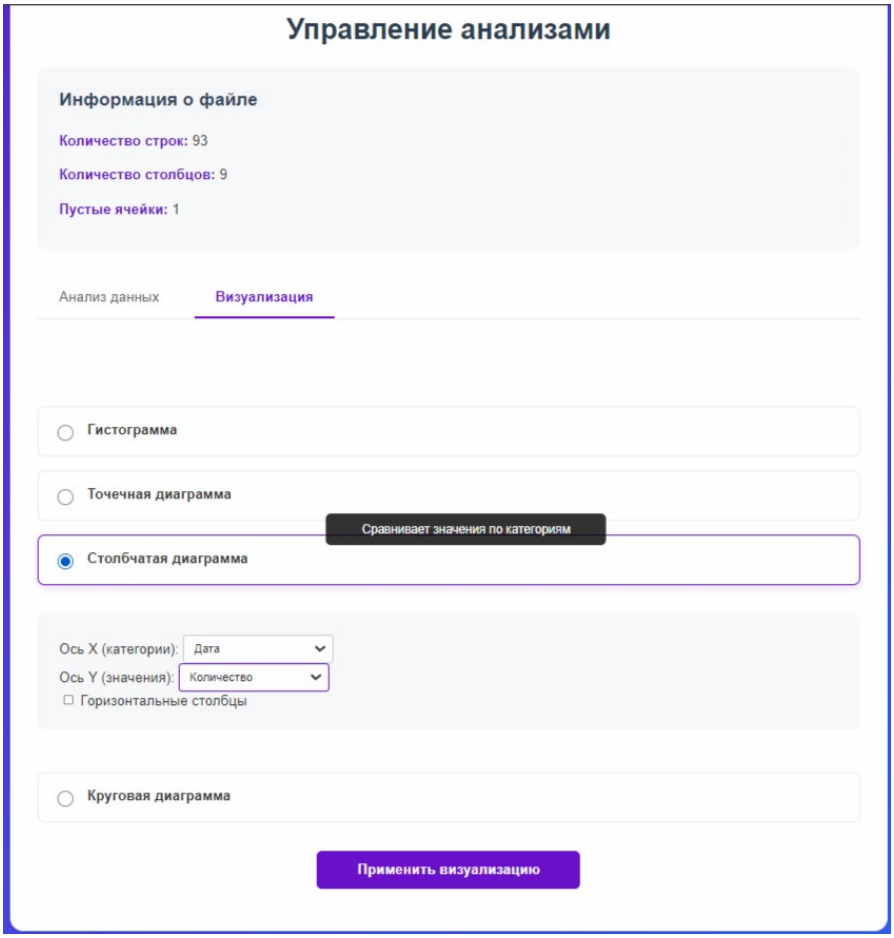


Рисунок 55 – Страница управления. Раздел визуализации

В данном разделе работает все аналогично разделу анализа данных. Выбираем шаблон «Столбчатой диаграммы» и задаем параметры для оси X и Y. Для этого отлично в загруженном файле подходим столбцы Даты и Количества соответственно, таким образом по результатам должен будет отобразиться график количества продаж за каждую неделю. После установки параметров, была нажата кнопка «Применить визуализацию». На рисунке 56 видно, что визуализация успешно была сгенерирована и отображена пользователю в соответствии с заданными им в меню управления параметрами.

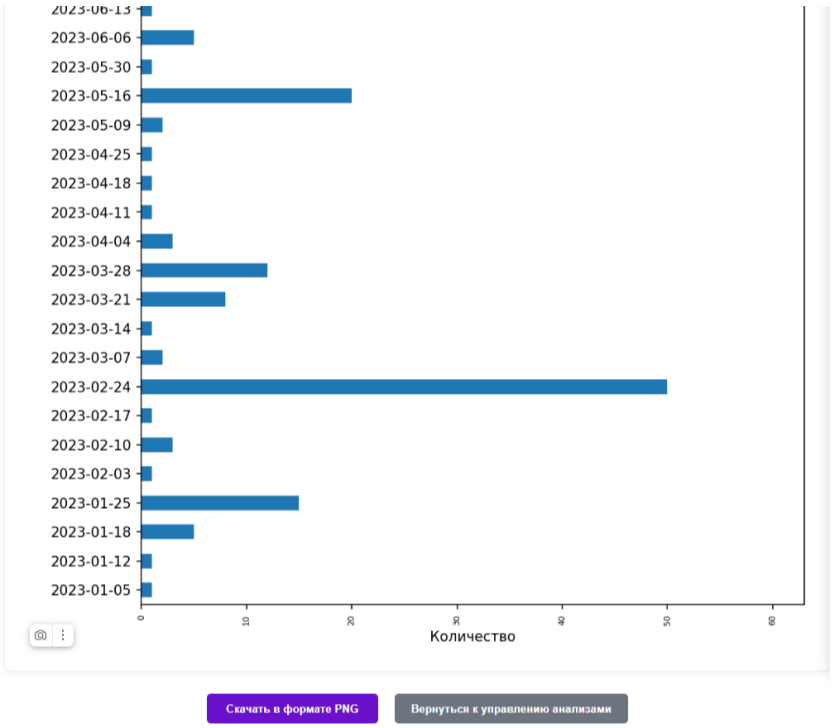


Рисунок 56 – Результаты визуализации по шаблону столбчатой диаграммы

После успешного создания визуализации, испытаем возможность скачивания результатов. Сделаем это воспользовавшийся прямой опции скачивания странице просмотров результата. На рисунке 57 показаны последствия нажатия кнопки скачивания – файл успешно перешел в загрузки.

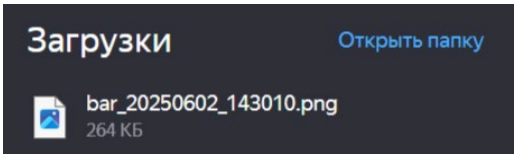


Рисунок 57 – Скачанный файл результата визуализации

Файл результатов визуализации в формате PNG был загружен на локальное устройство с соответствующий названием выбранному типу визуализации, времени его создания и формату.

Перейдем к демонстрации последней функциональной возможности в рамках контрольного примера – личному кабинету, отображению результатов в нем и возможности скачивать их оттуда. На рисунке 58 представлена страница личного кабинета пользователя.

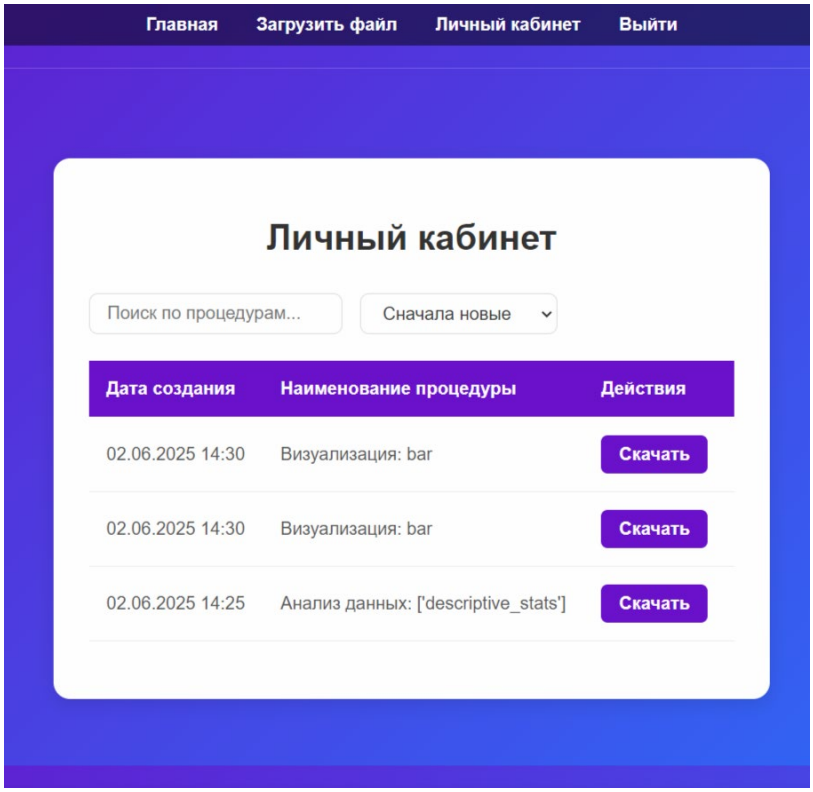


Рисунок 58 - Личный кабинет пользователя

На рисунке 58 видно, что последние созданные результаты по шаблонам на аккаунте успешно сохранились и отображаются в личном кабинете. Протестируем возможность скачивания результатов анализа, раннее созданных, напрямую из личного кабинета, нажав напротив него кнопку скачивания. Файл с результатами анализа при нажатии кнопки успешно переходит загрузки, как показано на рисунке 59.

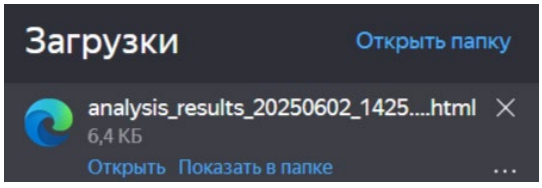
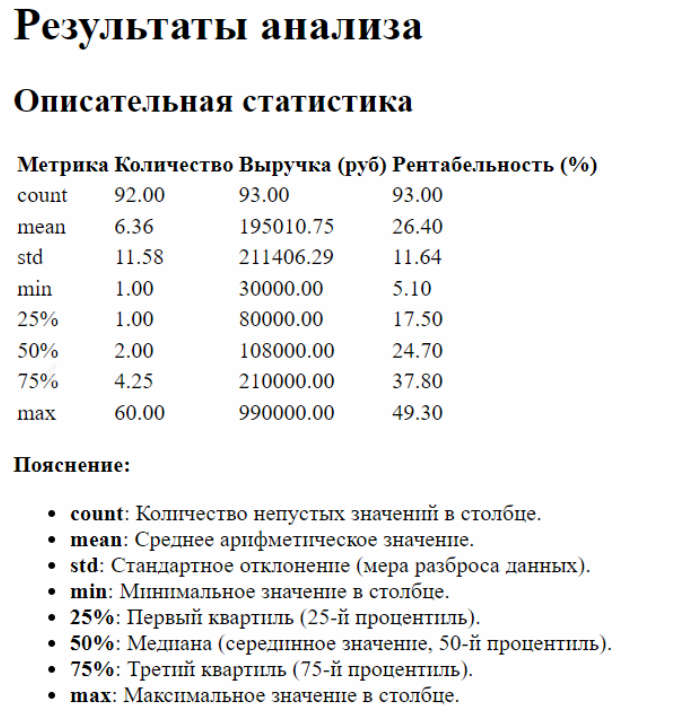


Рисунок 59 - Файл результатов анализа данных в загрузках

Результаты со соответствующим названием были успешно скачаны на пользовательское устройство. Откроем файл и проверим его содержание (рисунок 60):



Результаты анализа

Описательная статистика

Метрика	Количество	Выручка (руб)	Рентабельность (%)
count	92.00	93.00	93.00
mean	6.36	195010.75	26.40
std	11.58	211406.29	11.64
min	1.00	30000.00	5.10
25%	1.00	80000.00	17.50
50%	2.00	108000.00	24.70
75%	4.25	210000.00	37.80
max	60.00	990000.00	49.30

Пояснение:

- **count**: Количество непустых значений в столбце.
- **mean**: Среднее арифметическое значение.
- **std**: Стандартное отклонение (мера разброса данных).
- **min**: Минимальное значение в столбце.
- **25%**: Первый квартиль (25-й процентиль).
- **50%**: Медиана (серединное значение, 50-й процентиль).
- **75%**: Третий квартиль (75-й процентиль).
- **max**: Максимальное значение в столбце.

Рисунок 60 - Содержание файла скачанного результата анализа

На рисунке 60 можно увидеть, что результаты полностью соответствуют раннее сгенерированным и отображенным на странице просмотра результатов веб-приложения.

На этом контрольный пример и демонстрацию можно считать завершенной. Описанный в разделе пользовательский путь охватил работу всех реализованных модулей приложения, и все показали свою успешную работу и наглядно отразили возможности функционала и интерфейсы разработанного веб-приложения.

3.4 Оценка и обоснование экономической эффективности разработки

В данном разделе проводится оценка экономической эффективности внедрения веб-приложения для анализа и визуализации данных. Для расчета эффективности используется методика сравнения базового (существующего) и проектного (внедряемого) вариантов. Основное внимание уделяется снижению трудовых и стоимостных затрат, а также расчету срока окупаемости проекта.

Для оценки экономической эффективности выбрана методика расчета прямого эффекта [17], которая включает:

– Сравнение трудовых затрат (время выполнения операций);
– Сравнение стоимостных затрат (затраты на обработку информации);
– Расчет срока окупаемости проекта;
– Расчет трудовых затрат.

Рассчитывается трудоемкость базового варианта (T_0):

– Загрузка данных: 1 час;
– Анализ данных: 3 часа.;
– Визуализация данных: 2 часа;
– Сохранение и поиск результатов: 1 час.

Итого: $T_0 = 7$ часов.

Рассчитывается трудоемкость проектного варианта (T_1):

– Загрузка данных: 0,5 часа;
– Анализ данных: 1 час;
– Визуализация данных: 0,5 часа;
– Сохранение и поиск результатов: 0,2 часа.

Итого: $T_1 = 1{,}1$ часа.

Абсолютное снижение трудовых затрат (ΔT):

$$\Delta T = T_0 - T_1 = 3{,}5 - 1{,}1 = 2{,}4 \text{ часа}$$

Коэффициент относительного снижения трудовых затрат (K_T):

$$K_T = \left(\frac{\Delta T}{T_0}\right) * 100\% = \left(\frac{4{,}8}{7}\right) * 100\% \approx 68{,}6\%$$

Индекс снижения трудовых затрат (Y_T):

$$Y_T = \frac{T_0}{T_1} = \frac{3{,}5}{1{,}1} \approx 3{,}18$$

Рассчитываются стоимостные затраты. Стоимость базового варианта (C_0):

Затраты на оплату труда специалиста: 1000 руб./час.

Итого: $C_0 = 3{,}5 * 1000 = 3500$ руб.

Стоимость проектного варианта (C_1):

Итого: $C_1 = 1{,}1 * 1000 = 1100$ руб

Абсолютное снижение стоимостных затрат (ΔC):

$$\Delta C = C_0 - C_1 = 3500 - 1100 = 2400 \text{ руб.}$$

Коэффициент относительного снижения стоимостных затрат (K_C):

$$K_C = \left(\frac{\Delta C}{C_0}\right) * 100\% = \left(\frac{2400}{3500}\right) * 100\% \approx 68{,}6\%$$

Индекс снижения стоимостных затрат (Y_C):

$$Y_C = \frac{C_0}{C_1} = \frac{3500}{1100} \approx 3{,}18$$

Рассчитывается срок окупаемости проекта. Капитальные затраты на создание проекта ($K_П$):

Разработка программного обеспечения: 200 000 руб.

Затраты на внедрение и обучение: 50 000 руб.

Итого: $K_П$ = 250 000 руб.

Срок окупаемости (T_{OK}):

$$T_{OK} = \frac{K_П}{\Delta C} = \frac{250000}{2400} = 104 \text{ дня}$$

Все рассчитанные показатели эффективности были занесены и систематизированы в одной таблице 8.

Таблица 8 - Показатели эффективности

Показатель	КАК-ЕСТЬ вариант	КАК-ДОЛЖНО-БЫТЬ вариант	Абсолютное изменение	Коэффициент изменения	Индекс изменения
Трудоемкость(часы)	3,5	1,2	2,4	68,6%	3,18
Стоимость(рубли)	3500	1100	2400	68,6%	3,18

Подытожим проведенный экономический расчет. Использование веб-приложения для анализа данных и визуализации значительно ускоряет этот процесс, тем самым снижая стоимостные затраты на 68,6% от прошлого варианта. Индекс снижения затрат (Y_T и Y_C) составляет 3,18, что свидетельствует о значительном повышении производительности труда. По рассчитанному сроку окупаемости – проект окупится за 104 дня, но данное число очень относительно, учитывая, что объемы информации всегда будут разниться, несмотря на то, что внутри компании часто приходится воспроизводить процессы анализа данных и визуализации.

Тем не менее, решение с веб-приложением в любом случае выгоднее, так как как помимо экономической эффективности, так и несет множество качественных улучшений.

Выводы главы 3.

В третьей главе выпускной квалификационной работы было проведено тестирование функциональности разработанного веб-приложения, включая модульное тестирование и проверку сценариев использования. Для тестирования безопасности приложения были применены статический и динамические анализаторы (Bandit и OWASP ZAP соответственно), которые подтвердили соответствия требованиям защищенности. Работа системы была продемонстрирована на контрольном примере, охватывающем полный цикл использования: от регистрации пользователя до выполнения анализов и визуализаций данных, и скачивание их результатов. Так же был выполнен расчет экономической эффективности внедрения решения, показавший снижения трудозатрат и срок окупаемости в примерно 104 дня.

Заключение

По итогам выполнения работы была выполнена главная цель – разработано веб-приложение для анализа и визуализации данных. Оно отвечает современным вызовам цифровой трансформации бизнеса. В условиях стремительного роста объемов информации и необходимости оперативного принятия управленческих решений, традиционные подходы к обработке данных демонстрируют свою неэффективность. Использование разрозненных инструментов приводит к значительным временным затратам, рискам ошибок при переносе данных и необходимости дополнительного обучения сотрудников. Коммерческие BI-решения, в свою очередь, часто оказываются избыточными по функционалу и дорогостоящими для малых и средних предприятий.

Проведенное исследование рынка существующих решений выявило ключевые проблемы: отсутствие встроенных возможностей для глубокого статистического анализа, привязка к облачным сервисам с ограниченными возможностями кастомизации, а также сложности с интеграцией в существующую ИТ-инфраструктуру предприятий. Эти ограничения стали основой для формулирования требований к разрабатываемому решению.

Архитектура приложения строилась на принципах гибкости и масштабируемости. Выбор Flask в качестве основного фреймворка позволил создать легковесное, но мощное ядро системы, способное эффективно обрабатывать бизнес-логику. Интеграция с PostgreSQL обеспечила надежное хранение и быстрый доступ к метаданным, а использование Pandas и Matplotlib гарантировало точность аналитических расчетов и качество визуализации.

Процесс проектирования включал создание UML-диаграмм, детализирующих структуру системы и взаимодействие ее компонентов, разработку модели базы данных, а также применение методологии FURPS+ для управления требованиями. Это позволило создать целостное видение системы еще на этапе проектирования и избежать многих проблем на стадии реализации.

Реализованный функционал охватывает весь цикл работы с данными - от загрузки и валидации CSV-файлов до выполнения сложного статистического анализа и генерации интерактивных отчетов. Особое внимание было уделено

реализации мер безопасности и созданию интуитивно понятного интерфейса с персональным кабинетом пользователя, что значительно снижает порог вхождения для новых сотрудников и защищает пользовательские данные от возможного постороннего вмешательства.

Комплексное тестирование системы подтвердило ее стабильность и производительность. Автоматизированные тесты, выполненные с использованием pytest, охватили все критические компоненты системы, а ручное тестирование позволило оценить удобство работы с интерфейсом. Выявленные в процессе тестирования проблемы были оперативно устранены, что обеспечило высокое качество конечного продукта.

Главными преимуществами разработанного решения стали его экономическая эффективность, достигаемая за счет использования технологий с открытым исходным кодом, возможность развертывания на собственной инфраструктуре предприятия, что значительно повышает уровень безопасности данных, а также адаптивность системы, позволяющая легко расширять ее функциональность в соответствии с меняющимися бизнес-потребностями.

Перспективы развития системы связаны с расширением поддерживаемых форматов данных, внедрением алгоритмов машинного обучения для прогнозной аналитики и оптимизацией работы с большими объемами информации. Разработанное приложение не только решает актуальные проблемы предприятий в области анализа данных, но и создает основу для их цифровой трансформации, предлагая современный, эффективный и доступный инструмент для принятия управленческих решений.

Список используемой литературы и используемых источников

1. Brown, A. Low-Code and No-Code Platforms for Business Intelligence: Adoption Trends and Challenges / A. Brown, K. Davis // IEEE Software. – 2024. – Vol. 41, No. 2. – P. 45–52.

2. Chen, L. Security Best Practices in Modern Web Application Development Using Python Frameworks / L. Chen, H. Wang // ACM Computing Surveys. – 2023. – Vol. 55, No. 8. – P. 1–38.

3. Fowler, M. Patterns of Enterprise Application Architecture : учебное пособие / M. Fowler. – Boston : Addison-Wesley, 2022. – 560 p. – ISBN 978-0-13-683515-3.

4. FURPS+ Model Overview : техническая документация. – IBM, 2023. – URL: https://www.ibm.com/docs/en/SS9UMF_2025.01?topic=requirements-furps (дата обращения: 03.06.2025).

5. Gartner Market Guide for Analytics and Business Intelligence Platforms, 2023 : аналитический отчет. – URL: https://www.gartner.com/en/documents/4008017 (дата обращения: 03.06.2025).

6. OWASP Top Ten : руководство по безопасности. – URL: https://owasp.org/www-project-top-ten/ (дата обращения: 03.06.2025).

7. Richardson, C. Microservices Patterns: With examples in Java : учебное пособие / C. Richardson. – Shelter Island, NY : Manning Publications, 2023. – 520 p. – ISBN 978-1-61729-454-9.

8. Гринберг, М. Веб-разработка с применением Flask на языке Python : учебное пособие / М. Гринберг, А. Ганьо. – Санкт-Петербург : Питер, 2021. – 304 с. – ISBN 978-5-4461-1482-3.

9. Иванов, С. В. Безопасность веб-приложений: практические рекомендации : учебное пособие / С. В. Иванов. – Москва : ИНТУИТ, 2020. – 416 с. – ISBN 978-5-9963-4892-7.

10. Петрова Алина Витальевна Оптимизация бизнес-процессов как механизм обеспечения эффективного функционирования и развития предприятия // Скиф. 2023. №11 (87). URL: https://cyberleninka.ru/article/n/optimizatsiya-biznes-protsessov-kak-mehanizm-

obespecheniya-effektivnogo-funktsionirovaniya-i-razvitiya-predpriyatiya (дата обращения: 08.06.2025).

11. Козлов, И. С. Современные подходы к веб-аналитике в корпоративных системах / И. С. Козлов // Прикладная информатика. – 2022. – Т. 17, № 3. – С. 78–92.

12. Маккинни, У. Python и анализ данных : учебное пособие / У. Маккинни. – Москва : ДМК Пресс, 2023. – 482 с. – ISBN 978-5-93700-107-9.

13. Официальная документация Flask : техническая документация. – URL: https://flask.palletsprojects.com/ (дата обращения: 03.06.2025).

14. Официальная документация PostgreSQL : техническая документация. – URL: https://www.postgresql.org/docs/ (дата обращения: 03.06.2025).

15. Официальная документация библиотеки pandas : техническая документация. – URL: https://pandas.pydata.org/docs/ (дата обращения: 03.06.2025).

16. Рамальо, Л. Python. К вершинам мастерства : учебное пособие / Л. Рамальо. – Москва : ДМК Пресс, 2022. – 768 с. – ISBN 978-5-97060-845-1.

17. Смирнова, Г. Н. Экономическая эффективность информационных систем: методы оценки : учебное пособие / Г. Н. Смирнова. – Москва : Финансы и статистика, 2021. – 320 с. – ISBN 978-5-279-03489-1.

18. Федоров, Д. Ю. Микросервисная архитектура: принципы и практика : учебное пособие / Д. Ю. Федоров. – Москва : Издательские решения, 2022. – 214 с. – ISBN 978-5-4498-7890-1.

19. Хантер, Д. Matplotlib. Визуализация данных на Python : учебное пособие / Д. Хантер. – Санкт-Петербург : БХВ-Петербург, 2021. – 368 с. – ISBN 978-5-9775-3808-2.

20. Хог, Д. Д. PostgreSQL. Основы языка SQL : учебное пособие / Д. Д. Хог, Э. В. Варрен. – Москва : Символ-Плюс, 2020. – 496 с. – ISBN 978-5-6040724-5-9.