

## **ВВЕДЕНИЕ**

В последнее время под информационными технологиями чаще всего понимают компьютерные технологии. В частности, информационные технологии имеют дело с использованием компьютеров и программного обеспечения для хранения, преобразования, защиты, обработки, передачи и получения информации. Современные технологии превратили телефон из простого средства связи, коим он являлся несколько десятилетий назад, в персональный компьютер, помещающийся на ладони. Подобная метаморфоза стала возможной только благодаря научно-техническому прогрессу, сумевшему миниатюризировать свои достижения до размеров спичечного коробка, а также сделать их более доступными. В частности, наиболее популярная и используемая в мире операционная система смартфона является андроид.

Личные расходы – это ежемесячный индекс, отражающий изменения расходования средств для удовлетворения личных потребностей. Ведение такого учета позволяет оптимизировать и контролировать затраты, планировать распределение денежных средств для большей эффективности.

Для эффективного отслеживания затрат, как правило, ориентируются на чеки, являющиеся неотъемлемым атрибутом любых продаж юридического лица. Более чем 95% расходов денежных средств будет зафиксировано на чеках или квитанциях.

Таким образом допускается, что наилучший способ внести документ об оплате будет при его получении. В этом, несомненно, лучшим средством является смартфон.

Исходя из описанных данных можно утверждать о необходимости мобильного приложения, взаимодействующего с Интернет-системой. Web-клиент, в свою очередь, можно использовать на любом устройстве с любой операционной системой при наличии доступа ко всемирной паутине и программного обеспечения, позволяющего его осуществить, что также делает продукт более эффективным и распространенным.

Исходя из вышесказанного тема дипломного проекта – «Система для учета и анализа личных расходов», являющаяся актуальной в современном мире. Целью дипломного проектирования является программное обеспечение, которое можно использовать на любом устройстве, имеющем веб-браузер и выход в сеть Интернет.

Пояснительная записка состоит из восьми разделов, содержащих необходимую информацию по организации и эксплуатации программного средства.

Первый раздел «Объектно-ориентированный анализ и проектирование системы» содержит описание предметной области данного дипломного проекта, определение круга задач, которые необходимо автоматизировать, а также описание языка моделирования, используемого для разработки объектно-ориентированных систем.

Второй раздел «Вычислительная система» описывает минимальные и оптимальные конфигурации технических средств, обеспечивающих эффективное функционирование задачи; краткая характеристика выбранных программных средств, их достоинства; характеристика операционной системы, для которой разрабатывается программа.

Третий раздел «Проектирование задачи» включает основные требования, решения по архитектуре, пользовательские ресурсные файлы, структура входных и выходных данных, описание алгоритмов и прочих проектных решений.

Четвертый раздел «Описание программного средства» содержит обозначение и наименование приложения с указанием размера исполняемого и вспомогательных модулей; описание процесса инсталляции программы, функционального назначения, а также входных и выходных данных.

Пятый раздел «Методика испытаний» включает описание проверки каждого пункта меню, каждой операции, моделирование всех возможных действий пользователя при работе с программой.

Шестой раздел «Применение» включает описание назначения

программы. Кроме этого описываются средства защиты и структура справочной системы.

Седьмой раздел «Охрана труда» описывает мероприятия по технике безопасности. В частности, о санитарно-гигиенических требованиях к помещениям с персональными электронными вычислительными машинами и видеодисплейными терминалами.

Восьмой раздел «Экономический раздел» содержит расчеты для определения общей стоимости программного продукта, основной и дополнительной заработной платы, стоимостные затраты на материалы и прочие потребности.

В заключении будут подведены итоги дипломного проектирования. Описаны достоинства, недостатки и особенности разработанного программного продукта.

# **1 ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ СИСТЕМЫ**

## **1.1 Сущность задачи**

Тема дипломного проекта – разработка «Система для учета и анализа личных расходов». На основании описания предметной области необходимо спроектировать архитектуру программы и разработать готовый продукт для работы с личными затратами.

Актуальность дипломного проекта заключается в учете сведений о потраченных средствах. Автоматизированный учет позволит повысить эффективность управления финансами, снизить временные затраты по подсчету и записям, вести учет чековой информации, а также формировать отчетную документацию.

Объектом исследования дипломного проекта является дневник личных расходов.

Предметом исследования представляются методы и алгоритмы для реализации автоматического программного продукта, взаимодействующего с сервером и базой данных.

Целью дипломного проекта заключается написание ряда программного обеспечения, которое позволит вести полный учет расходов, минимизируя затраты времени.

Главные задачи данного дипломного проекта следующие:

- 1) исследовать предметную область, выделить сущности, атрибуты и связи между ними;
- 2) определить требования к разрабатываемому программному продукту;
- 3) реализовать паттерны проектирования, в ходе которых будут получены модели данных;
- 4) выполнить функциональное и полное тестирования.

Техническое задание составлено для разработки программного комплекса, составляющего собственный учет расходов при помощи метода распознавания графических образов.

Требования к разработке:

- 1) пользовательский интерфейс должен быть прост, удобен и доступен неподготовленному пользователю;
- 2) необходимо обеспечивать выполнение всех эргономических требований (комфортность, цветовую и звуковую гамму, соответствующие наилучшему восприятию, удобство расположения информации и доступность всех необходимых для работы средств, единый стиль выполнения операций и т.д.).

Цель работы заключается в верно составленном подходе решения поставленной задачи, разработке и отладке программы, реализующей требуемый функционал, исходя из данных технического задания.

Предметная область представлена следующей информацией: разработать систему, которая позволяет обрабатывать информацию о расходах.

Реализовать возможности:

- ручного ввода данных;
- обработку фотографий;
- добавления, удаления, редактирования чековых данных;
- получения и анализа данных на критерии выбора диапазона дат;
- отображения диаграмм;
- экспорта данных в MS Word и MS Excel.

При разработке ориентироваться на постановку задач и описание предметной области.

## **1.2 Проектирование модели**

Проектирование программного обеспечения – этап жизненного цикла программного обеспечения, во время которого исследуется структура и

взаимосвязи элементов разрабатываемой системы. Результатом этого этапа является прежде всего набор документов, содержащий в себе достаточное количество информации для реализации системы.

На этапе проектирования уточняется функциональная спецификация системы: прорабатывается архитектура системы, определяются требования к аппаратному обеспечению. Также определяется набор организационных мероприятий, необходимых для внедрения системы, и перечень документов, регламентирующих ее использование.

При проектировании системы использованы следующие схемы решений, влияющих на архитектуру системы:

- 1) принцип открытости;
- 2) принцип стандартизации (унификации);
- 3) принцип преемственности.

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования объектно-ориентированных систем. UML призван поддерживать процесс моделирования программного средства, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла программного средства, начиная с бизнес-анализа и заканчивая сопровождением системы.

Данный подраздел описывает четыре диаграммы:

- 1) деятельности;
- 2) развертывания;
- 3) вариантов использования;
- 4) последовательности действий.

Диаграмма деятельности – UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов – вложенных видов деятельности и отдельных

действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмме, изображённая на рисунке В.1, состоит из следующих действий:

- 1) выбор графического файла;
- 2) выполнение снимка;
- 3) отображение фотографии;
- 4) отправление файла на сервер;
- 5) распознавание данных;
- 6) обработка данных;
- 7) сохранение данных в базе данных.

Также на ней отображены блоки на проверку истинности условий и три конечных узла, завершающих все потоки диаграммы: «переход к авторизации», «отмена операции» и «чек внесен». Таким образом, была спроектированная диаграмма деятельности.

Диаграмма развертывания (deployment diagram) – диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

На рисунке В.2, отображена диаграмма развёртывания. Диаграмма, включает в себя следующие узлы:

- 1) «android-клиент»;
- 2) «рабочая станция»;
- 3) «сервер сайта»;
- 4) «web browser»;
- 5) «сеть»;
- 6) «база данных».

Основным назначением методологии SADT является моделирование предметной области с целью определения требований к разрабатываемой системе или программному средству и с целью их проектирования. Методология SADT может применяться при выполнении ранних работ процесса разработки

системы или программного средства.

Диаграмма вариантов использования описывает организацию поведения системы: совокупность прецедентов и актеров, а также отношения между ними. Иллюстрируется статический вид системы с точки зрения прецедентов, что необходимо для организации и моделирования поведения.

На листе 1 показана диаграмма вариантов использования, составленная для данной программы. В таблице 1 представлены и описаны объекты диаграммы.

Таблица 1 – Таблица объектов диаграммы претендентов

Актеры	Варианты использования	Отношения
Зарегистрированный пользователь	Редактирование чека	Отношения ассоциации
	Просмотр данных	
	Данные профиля	Отношения расширения и включения
	Детальная информация	
	Добавление данных чека	
	Ручное внесение данных чека	Отношения расширения
	Добавление визуального образа чека	
	Изменение данных профиля	
	Сравнение данных	
	Удаление чека	Отношения включения
	Изменение чека	
Незарегистрированный пользователь	Просмотр информации о сайте	Отношение ассоциации
	Авторизация	
	Регистрация	
	Добавление пользователя	Отношения включения

Таким образом, вариант использования описывает, с точки зрения действующего лица, группу действий в системе, которые приводят к конкретному результату. Варианты использования являются описаниями



типичных взаимодействий между пользователями системы и самой системой. Они отображают внешний интерфейс системы и указывают форму того, что система должна сделать.

Диаграммы последовательностей акцентируют внимание на временной упорядоченности сообщений, т.е. позволяет отслеживать поведение взаимодействующих групп объектов. На них изображают множество объектов и посланные или принятые ими сообщения. Объекты, как правило, представляют собой анонимные или именованные экземпляры классов, но могут быть также экземплярами других сущностей, таких как кооперации, компоненты или узлы.

На листе 2 показана диаграмма последовательности действий для отображения анализа данных на основе сравнения заданных временных рамок. Актер – пользователь – нажимает на кнопку, в следствии чего web-страница сравнения открывается. В поле выбора устанавливается значение ручного ввода, в следствии чего отображаются поля отправления формы, которые далее заполняются. После ввода нажимается кнопка «Сравнить» и возвращается представление отображения данных по заданному диапазону дат.

Указанные диаграммы являют собой достаточную базу для составления проектного решения, отражающего сущность задачи.

## 2 ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

«Система для учёта и анализа личных расходов» разработана под операционной системой семейства Linux на основе дистрибутива Ubuntu 18.04 LTS x64.

Операционная система Ubuntu 18.04 – это современная многозадачная многопользовательская 32- или 64-разрядная операционная система с графическим интерфейсом пользователя и приоритетной многозадачностью, основанная на Debian GNU/Linux.

Надежность и безопасность Ubuntu остается на высоком уровне в сравнении с другими современными операционными системами нового поколения. Характерны высокое качество, оптимизация и надежность. Данная система является первой в списке самых популярных дистрибутивов Linux для веб-серверов.

Операционные системы семейства Ubuntu являются наиболее распространенными на офисных компьютерах, связанных с информационными технологиями, а также среди научного сообщества.

Программа для смартфона на основе операционной системы Android разработана в программном средстве Android Studio 3.4 на версии JDK Java Development Kit 8 с использованием языка Java.

Android Studio – это интегрированная среда разработки для работы с платформой Android, основанная на IntelliJ IDEA.

К достоинствам относятся:

- среда разработки поддерживает работу с несколькими языками программирования, к которым относятся: C/C++, Kotlin, Java.
- редактор кода;
- позволяет разрабатывать приложения для смартфонов/планшетов, а также для портативных ПК, приставок телевизоров Android TV, устройств Android Wear, устройств с необычным соотношением сторон экрана;

- тестирование корректности работы утилит, их производительности на той или иной системе, происходит непосредственно в эмуляторе;
- рефакторинг готового кода;
- большая библиотека с готовыми шаблонами и компонентами для разработки программного обеспечения;
- предварительная проверка уже созданного приложения на предмет ошибок в нем;
- большой набор средств инструментов для тестирования каждого элемента приложения;
- для неопытных/начинающих разработчиков специально создано руководство по использованию Android Studio, размещенное на официальном сайте утилиты.

Недостатки заключаются в высоких требованиях к устройству по производительности аппаратной основы, на котором планируется тестирование, в том числе невозможность написать серверные проекты на языке Java.

Программирование на языке Java представляет собой перспективное направление, которое позволяет реализовать множество современных подходов программирования.

Java представляет собой язык программирования и платформу вычислений. Существует множество приложений и веб-сайтов, которые не работают при отсутствии установленной Java, и с каждым днем число таких веб-сайтов и приложений увеличивается. Java отличается быстротой, высоким уровнем защиты и надежностью. От портативных компьютеров до центров данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет.

Web-сервер и -клиент разработаны в программном средстве PyCharm 2019.1 с использованием языка Python.

PyCharm – это интеллектуальная интегрированная среда разработки с полным набором средств для эффективной разработки на языке Python.

Python – высокоуровневый язык программирования общего назначения,

ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен, поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

PyCharm – это интеллектуальная интегрированная среда разработки с полным набором средств для эффективной разработки на языке Python.

К достоинствам можно отнести «умный» редактор кода, навигации. Предоставляются встроенные инструменты разработки:

- а) профайлер Python;
- б) встроенный терминал;
- в) интеграцию с большими VCS и встроенными инструментами баз данных;
- г) возможность удаленной разработки с удаленными интерпретаторами;
- д) интегрированный терминал ssh;
- е) интеграция с Docker и Vagrant.

Также предоставление отладки, тестирования, профилирования, инструменты баз данных, научные инструменты, веб разработка, возможность использования плагинов.

Поддержка базы данных осуществлена в программном средстве DataGrip 2019.1 с использованием PostgreSQL.

Некоторые из предоставленных возможностей:

- лог всех запросов;
- поддержка большинства известных систем управления базами данных;
- формater кода;

- запуск хранимых процедур;
- миграция исходников;
- план запроса в виде диаграммы;
- редактирование SQL;
- редактор данных;
- улучшения в пользовательском интерфейсе.

Программное обеспечение JetBrains DataGrip представляет собой интегрированную среду разработки для работы с базами данных MySQL, PostgreSQL, Oracle, SQL Server, Sybase, DB2, SQLite, HyperSQL, Apache Derby и H2. Решение включает текстовый редактор с мультикурсорами, обеспечивает синтаксическое выделение кода, поддерживает интеграцию с системами контроля версий Git, Subversion.

Выбранные инструменты позволят реализовать и упростить программное средство «Система для учета и анализа личных расходов».

## **3 ПРОЕКТИРОВАНИЕ ЗАДАЧИ**

### **3.1 Требование к приложению**

В системе для учета и анализа личных расходов должна иметься возможность работы с чековыми данными, доступные для авторизованного пользователя:

- добавление данных посредством ручного ввода;
- обработка данных на основе фотографии чека;
- редактирование товара, включая его наименование, категорию, тип и цену;
- удаление товара из чека;
- единичное и групповое удаление чека;
- экспорт данных в форматы Word и Excel.

Также следует предусмотреть анализ данных при выборе периода на основе товаров и заработной платы, сравнение информации по промежуткам времени, построение диаграмм различных типов. Предоставить работу с учетной записью: смена пароля, выход из системы, изменение имени пользователя и установка заработной платы.

Функционал, доступный неавторизованному пользователю должен включать возможность регистрации, входа в систему, скачивание Android-приложения, предоставление информации ознакомительного характера и основные положения проекта.

Необходимо предусмотреть понятный и эстетический интерфейс. Должны отсутствовать раздражительные цвета, а выбранные – гармонизировать друг с другом, представления следует вести в одном стиле. Для создания полезного и удобного интерфейса в системе для учета и анализа личных расходов необходимо соблюдение требований:

- наличие удобного меню;

- кнопки должны быть подписаны;
- шрифты, используемые в программе, должны быть понятны;
- при работе у пользователя не должно возникать дискомфорта;
- все надписи изображены на русском языке.

Также должна присутствовать справочная система, позволяющая ознакомиться с возможностями и правилами использования, содержащих примеры работы с функциями программы. Для удобства её следует разделить на разделы и подразделы, чтобы пользователю было легче ориентироваться в ней.

Чтобы предотвратить большинство ошибок во время использования продукта, нужно использовать следующие проверки:

- а) при работе с полями ввода данных или выбора должна иметься проверка на пустое поле, при незаполнении – сообщение о необходимости внести информацию;
- б) проверка на тип значения, например, корректное выставление в поле даты;
- в) серверную валидацию на отправленные данные.

Для работы в режиме настоящего времени задействовать языки программирования JavaScript, JQuery и Ajax.

### **3.2 Концептуальный прототип**

Концептуальный прототип — это общее видение продукта, его функций, логики взаимодействия с пользователями. Подразумевается описание внешнего пользовательского интерфейса, а именно описание:

- а) системы пользовательского меню;
- б) элементов управления (различных компонентов, при помощи которых облегчается работа пользователя).

В системе для учета и анализа личных расходов будет использоваться навигационное меню, содержащее в левой части иконку приложения, пункт «Домашняя страница», перемещающая на соответствующую страницу. В правой

части для авторизованного пользователя будет отображено приветствие и «Профиль», осуществляющие переход на страницу учетной записи и «Выход», позволяющий выйти из системы. Для незарегистрированного пользователя вместо вышеописанных пунктов будут располагаться «Войти», перемещая на страницу авторизации, и «Зарегистрироваться», возвращая соответственное представление.

Для незарегистрированного пользователя доступны следующие кнопки:

- 1) «Зарегистрироваться» – создает учетную запись;
- 2) «Войти» – осуществляет вход в систему;
- 3) «Забыли пароль» – перенаправляет на страницу сброса пароля;
- 4) «Сбросить пароль» – отправляет на почтовый адрес пользователя ссылку для сброса пароля;
- 5) «Скачать» – загружает Android-клиент;
- 6) «Ознакомиться» – открывает страницу политики конфиденциальности;
- 7) «Пользовательское соглашение» – открывает страницу с основными положениями проекта;
- 8) «Вернуться на домашнюю страницу» – возвращает на начальное представление.

Для зарегистрированного пользователя доступны также следующие кнопки:

- 1) «В профиль» – открывает страницу учетной записи;
- 2) «Добавить чек» – перенаправляет на страницу добавления чека;
- 3) «Обработать» – распознает графические файлы и распределяет в чек;
- 4) «Внести» – формирует из заполненных полей данные чека;
- 5) «Добавить поле +» – добавляет поле для ввода продуктов;
- 6) «Очистить» – удаляет загруженные фотографии;
- 7) «Перейти к сравнению» – открывает соответственную страницу;
- 8) «Excel» – экспортирует чек / данные за период в формат .xls;
- 9) «Word» – экспортирует чек / данные за период в формат .docx;



- 10) «Удалить выделенные» – удаляет помеченные чеки;
- 11) «Удалить» – удаляет выбранный чек;
- 12) «Редактировать профиль» – открывает страницу изменения профиля;
- 13) «Подробная информация» – открывает соответствующее представление;
- 14) «Отправить» – изменяет данные профиля.

В данном программном средстве пользовательские роли не требуются.

### **3.3 Организация данных**

На сегодняшний день большая часть задач требует работы с большим количеством данных – это и получение, и обработка, и хранение информации. Существует множество различных продуктов для работы с базами данных, которые объединены общими принципами.

Организация данных – представление данных и управление данными в соответствии с определенными соглашениями.

Система управления базами данных (СУБД) — специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных. Для создания и управления информационной системой СУБД необходима в той же степени, как для разработки программы на алгоритмическом языке необходим транслятор.

Встраиваемая СУБД – библиотека, которая позволяет унифицированным образом хранить большие объёмы данных на локальной машине. Доступ к данным может происходить через геоинформационные системы.

PostgreSQL – это свободно распространяемая объектно-реляционная система управления базами данных. PostgreSQL разрабатывается международным сообществом разработчиков и не контролируется компаниями или частными лицами. Распространяется по свободной лицензии, которая позволяет включать его в состав коммерческих программных продуктов.

Проектирование структуры базы данных исходило из построенной

диаграммы «Сущность-связь». Эта схема выявляет основные объекты предметной области, значимые для решаемой задачи, показывает отношения между ними. Любой фрагмент предметной области может быть представлен как множество сущностей со множеством связей. Схема используется в качестве основы для унификации различных представлений на основе сетевой, реляционной и модели множества сущностей.

Схема «Сущность-связь» для системы по учету и анализу личных затрат представлена на изображении В.3, структура таблиц базы данных показана в таблицах 2-8.

Таблица 2 – Структурная таблица «user»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код пользователя
email	varchar(255)		Почтовый адрес пользователя
registred_on	timestamp		Дата-время регистрации
username	varchar(50)		Имя пользователя
password_hash	varchar(100)		Хэш пароля
public_id	varchar(100)		Публичный код пользователя
wage	numeric(15, 2)		Заработная плата
is_admin	boolean		Принадлежность к администрации
confirmed	boolean		Подтвержденность регистрации
confirmed_on	timestamp		Дата-время подтверждения регистрации

Таблица 3 – Структурная таблица «organization»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код организация

Окончание таблицы 3 – Структурная таблица «organization»

legal_name	varchar(50)		Название организации
legal_adress	varchar(50)		Адрес организации
taxpayer_ identification_number	integer		Уникальный номер предпринимателя

Таблица 4 – Структурная таблица «check»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код чека
*organization_id	integer		Код организации, вторичный ключ к таблице «organization»
*user_id	integer		Код пользователя, вторичный ключ к таблице «user»
date_time_of_ purchase	timestamp		Дата и время покупки

Таблица 5 – Структурная таблица «product»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код продукта
*check_id	integer		Код чека, вторичный ключ к таблице «check»
*abstract_product_id	integer		Код абстрактного продукта, вторичный ключ к таблице «abstract_product»
product_name	varchar(254)		Наименование продукта
product_price	numeric(15, 2)		Стоимость продукта

Таблица 6 – Структурная таблица «abstract\_product»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код абстрактного продукта
*product_category_id	integer		Код категории продукта, вторичный ключ к таблице «product_category»
product_name	varchar(80)		Наименование абстрактного продукта

Таблица 7 – Структурная таблица «product\_category»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код категории продукта
*category_type_id	integer		Код типа категории, вторичный ключ к таблице «category_type»
product_category_name	varchar(80)		Наименование категории продукта

Таблица 8 – Структурная таблица «category\_type»

Имя поля	Тип данных	Ключевое	Описание поля
#id	integer	+	Код вида платежа
category_type_name	varchar(80)		Наименование типа категории

Таким образом, была спроектирована модель базы данных, содержащая семь таблиц. Были определены первичные и вторичные ключи, атрибуты каждой таблицы и их типы данных, реляционная модель данных представлена иллюстрацией В.9.

### 3.4 Функции: логическая и физическая организация

Проект – это специальная структура управления, осуществляющая

взаимодействие между программным кодом и визуальными объектами.

Модуль представляет собой функционально законченный фрагмент программы

Оформленный в виде отдельного файла с исходным кодом или поименованной непрерывной его части, предназначенный для использования в программе. Модули позволяют разбивать сложные задачи на более мелкие. Система для учета и анализа личных расходов содержит в себе следующие основные модули:

- 1) «app.py» – содержит конфигурацию запуска сервера, являя собой точку входа в программное средство;
- 2) «config.py» – включает настройку конфигурационных данных;
- 3) «session.py» – модуль для работы с сессиями;
- 4) «site.css» – каскадная таблица стилей;
- 5) «base.html» – шаблон представлений;
- 6) «footer.html» – подвал в представлениях;
- 7) «profile.html» – представление профиля;
- 8) «edit\_profile.html» – представление редактирования профиля;
- 9) «detail\_information.html» – представление подробной информации;
- 10) «compare\_information.html» – представление сравнения данных;
- 11) «check\_details.html» – представление просмотра чека;
- 12) «add\_check.html» – представление добавления чека;
- 13) «index.html» – представление домашней страницы;
- 14) «about.html» – представление «О сайте»;
- 15) «500.html» – представление ошибки 500;
- 16) «activate.html» – представление html-шаблона при отправке сообщения на почту пользователя для подтверждения создания учетной записи;
- 17) «login.html» – представление входа в систему;
- 18) «register.html» – представление регистрации;
- 19) «confirm\_email.html» – представление подтверждения почты;

- 20) «reset\_password.html» – представление сброса пароля;
- 21) «authorization\_controller.py» – контроллер авторизации;
- 22) «home\_controller.py» – контроллер основного представления сайта;
- 23) «support\_controller.py» – контроллер справки;
- 24) «user\_controller.py» – контроллер пользователя;
- 25) «check\_controller.py» – контроллер, предоставляющий rest api чека;
- 26) «organization\_controller.py» – контроллер, предоставляющий rest api организации;
- 27) «product\_controller.py» – контроллер, предоставляющий rest api продуктов;
- 28) «export\_data.py» – модуль, отвечающий за экспорт данных в Word и Excel;
- 29) «AbbyyOnlineSdk.py» – модуль, предоставляемый Abbyy Cloud для предоставления взаимодействия с api распознавания;
- 30) «image\_processing.py» – модуль, для обработки фотографии;
- 31) «check\_parser.py» – модуль для извлечения данных чека;
- 32) «check\_processing.py» – модуль обработки чека;
- 33) «user.py» – модуль класса пользователя базы данных;
- 34) «product.py» – модуль класса продукта базы данных.

Существуют также менее важные модули, такие как для обработки данных даты, отвечающие за ошибочные ситуации, отправку сообщений на почтовый адрес и другие.

### **3.5 Функции и элементы управления**

В данном проекте было использовано большое количество элементов управления, таких как:

- 1) компонент «select», для вывода критериев диапазона дат в

представлениях:

- а) «compare\_information»;
  - б) «detail\_information».
- 2) компонент «table», для вывода чеков в представлении «profile»;
  - 3) компонент «button», для отправки данных формы, перенаправления на другие представления и прочие взаимодействия с пользователем;
  - 4) компонент «chart», для отображения диаграмм на страницах:
    - а) «compare\_information»;
    - б) «detail\_information».
  - 5) компонент «input», для ввода информации, среди которой заполнение данных продуктов, поля авторизации и другое;
  - 6) компонент «date», для выбора даты в представлениях:
    - а) «compare\_information»;
    - б) «check\_details»;
    - в) «add\_check»;
    - г) «detail\_information».

Также в системе используются компоненты «a», «p», «div», «br», «hr», «li», «ul», «ol», «h...» и другие.

### **3.6 Проектирование справочной системы приложения**

Справочная система будет разработана как отдельное представление, использующее заголовки для удобства пользователя в виде открытия и скрытия информации.

Будут описаны все функции системы, в частности, как выполнить то или иное действие. Также будет наличие экранной копии с примерами, следуя последовательному исполнению с описаниями.

Проектирование раздела помощи будет произведено в виде отдельного представления web-клиента.

## **4 ОПИСАНИЕ ПРОГРАММНОГО СРЕДСТВА**

### **4.1 Общие сведения**

Система для учета и анализа личных расходов была разработана в операционной системе Linux дистрибутива Ubuntu 18.04 LTS. В качестве интеграционной среды разработки использовался PyCharm и Android Studio.

Система занимает около 700 Мегабайт. Для работы необходимо наличие базы данных PostgreSQL. Достаточно пустой структуры, не заключающей в себе записи. Также требуется наличие всех используемых библиотек, установленных в системе или виртуальном окружении.

Чтобы запустить сервер достаточно открыть PyCharm и воспользоваться горячими клавишами «Shift» + «F10», либо произвести запуск посредством терминала, находясь в виртуальном окружении, следующей командой: «run app.py».

### **4.2 Функциональное назначение**

Основной функцией системы для учета и анализа личных затрат является автоматизация обработки чековой информации.

В системе предусмотрено выполнение следующих функций:

- 1) управление чековыми данными:
  - а) ручное добавление;
  - б) добавление посредством обработки фотографии (единичное и групповое);
  - в) редактирование в режиме прямого доступа;
  - г) удаление (единичное и групповое);
  - д) экспорт данных в форматы Word и Excel.
- 2) просмотр подробной информации по затратам в течении заданного периода;



- 3) сравнение данных на основе выбора периодов (как равнозначных, так и пользовательских);
- 4) предоставление информации о проекте и основных положениях;
- 5) полная система авторизации.

В программе присутствует два типа пользователя: зарегистрированный – обладают возможностью использовать весь функционал своей учетной записи, – и незарегистрированный – доступен только просмотр данных, имеющий непосредственное отношение к проекту.

Для возможности использования всего функционала необходимо зарегистрироваться, после чего пройти процедуру входа. Для этого требуется перейти на соответствующее представление, далее ввести логин и пароль – по желанию выставить поле «Запомнить меня» в значение правды, – после чего нажать на кнопку «Войти».

Чтобы зарегистрироваться, необходимо перейти на представление регистрации, после чего заполнить текстовые собственными данными. В завершение следует нажать на кнопку «Зарегистрироваться», после чего на адрес, указанный в поле «Email», будет выслана инструкция и ссылка, после перехода которой учетная запись будет активирована.

#### **4.3 Входные данные**

Входная информация – это информация, которая вводится пользователем и обрабатывается программой во время ее работы.

В программном средстве «Система для учета и анализа личных расходов» наибольшее количество входной информации осуществляется при вводе информации в текстовые поля, в том числе загрузка изображений. Также входной будут считаться те данные, которые загружаются в виде файлов и отправляются в виде post-запросов посредством api.

Перед тем, как входная информация добавится в базу данных, она проходят ряд проверок на корректность. Происходит проверка на то, чтобы все

необходимые поля были заполнены. Это позволяет избежать возникновения пустых записей. Также в программе происходит проверка на дублирование входных данных.

#### **4.4 Выходные данные**

Выходная информация – это информация, получаемая в ходе использования программного средства.

Результатом выходной информации программного средства «Система для учета и анализа личных расходов» является запись данных в файлы следующих форматов: «xml» – являют собой распознанный текст из загружаемых фотографий чеков, «docx» и «xls» – представляют экспорт данных по чеку или за указанный период; каждое из web-представлений.

К выходной информации относятся диаграммы, отображаемые в детальной информации за указанные периоды времени, среди которых:

- 1) «Pie» – круговая диаграмма, отображающая категории продукции;
- 2) «Line» – указывает на финансовые затраты, разделенные на секции в выбранном периоде;
- 3) «Radar» – диаграмма, предоставляющая разницу между двумя периодами в затратах по категориям продукции;
- 4) «Bar» – указывает на разницу в категориях продукции, разделенных на секции за два выбранных периода;
- 5) «Circle» – круговая диаграмма с внутренним вырезом, отображающая типы продукции;
- 6) круговые диаграммы, созданные внутри экспортных данных файла формата «xls».

Кроме этого, выходной информацией являются сообщения с вопрошанием подтверждения выполняемых операций, а также данные базы данных, в том числе возвращаемый через api в формате json.

## **5 МЕТОДИКА ИСПЫТАНИЙ**

### **5.1 Технические требования**

Для стабильной работы системы для учета и анализа личных расходов необходим современный компьютер, а также стабильное Интернет-соединение, прилагающаяся к устройству мышь и клавиатура. В том числе в рабочей системе должен присутствовать современный web-браузер:

- Google Chrome;
- Opera;
- Microsoft Edge;
- Safari;
- IE 9.0+;
- Mozilla Firefox.

Таким образом, программное средство «Система для учета и анализа личных расходов» является нетребовательным веб-приложением.

### **5.2 Порядок проведения испытаний**

Тестирование программного обеспечения – это процесс выявления ошибок в программном средстве. Существующие на сегодняшний день методы тестирования программного обеспечения не позволяют однозначно и полностью устранить все дефекты и ошибки и установить корректность функционирования анализируемой программы особенно в закрытых частных программах. Поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО.

Такой процесс формальной проверки или верификации может доказать, что дефекты отсутствуют, с точки зрения используемого метода.

Тестирование проводится с целью обеспечить качество разрабатываемого программного продукта. Стандарт ISO-8402, посвященный описанию систем

обеспечения качества программного обеспечения, под качеством понимает «совокупность характеристик программного продукта, относящихся к его способности удовлетворять установленные и предполагаемые потребности клиента». Основным параметром качества программы является надёжность. Надёжность определяется как вероятность его работы без отказов в течении определённого периода времени, рассчитанная с учётом стоимости для пользователя каждого отказа. Отказ программного обеспечения - это проявление ошибки в нём. Отсюда тестирование программного обеспечения – это процесс выполнения программы с целью обнаружения в ней ошибок.

Тестирование программного обеспечения – это попытка определить, выполняет ли программа то, что от неё ожидают. Качество программных средств можно определить, как совокупную характеристику исследуемого программного обеспечения с учётом следующих составляющих:

- 1) надёжность;
- 2) сопровождаемость;
- 3) практичность;
- 4) эффективность;
- 5) мобильность;
- 6) функциональность.

Существует множество подходов к решению задачи тестирования и верификации ПО.

Таблица 9 – Тестовые проверки

№	Тест	Глубина тестового покрытия	Результат теста
1	Все параграфы, заголовки, статьи не содержат ошибок	МАТ	Пройден
2	Контактные данные точны	МАТ	Не пройден
3	Картинки и анимация на нужных местах и не выходят за рамки шаблона	МАТ	Пройден
4	Сайт корректно отображается в любом браузере	АТ	Пройден
5	Сайт корректно отображается на всех портативных девайсах	АТ	Пройден

6	Изображения оптимизированы на всех страницах	АТ	Пройден
7	Стиль абзацев выдержан	АТ	Пройден
8	Формы для предоставления данных верны и работают	Smoke	Пройден
9	Внутренние ссылки работают на всех страницах	Smoke	Пройден
10	Внешние ссылки работают на всех страницах и открываются в новых вкладках	Smoke	Пройден
11	Время загрузки каждой страницы оптимизировано	АТ	Пройден
12	Каждая страница имеет уникальный URL	МАТ	Пройден
13	Правильная навигация сайта	АТ	Пройден
14	Установлен и настроен счётчик посещений liveinternet.ru	АТ	Не пройден
15	Все страницы сайта понятны и просты в использовании	МАТ	Пройден
16	Доступ к главному меню осуществляется со всех страниц	МАТ	Пройден
17	Корректная работа сайта на разных расширениях экрана	АТ	Не пройден
18	Скорость загрузки сайта, в условиях ограниченной мощности ПК	МАТ	Пройден

Схема тестирования программного средства представлена на изображении В.6, которая демонстрирует часть проведенного тестирования.

### 5.2.1 Функциональное тестирование

Запустили веб-приложение в браузере, после открылась главная страница сайта, как показано на изображении Б.1.

Перешли на страницу «О сайте» по нажатию на соответствующий пункт в меню, расположенный в верхней части каждого представления. Результат отображения представлен на рисунке Б.2.

Нажали «Зарегистрироваться» в правой части навигационного меню, как показано на рисунке Б.3. Открылась форма регистрации, как указано на изображении Б.4. Заполнили данные, как на картинке Б.5, после чего нажали кнопку «Зарегистрироваться», которая видна на изображении Б.4. Открыли почтовый адрес, указанный при регистрации, на который пришло письмо с

подтверждением регистрации на подобии указанного в примере на рисунке Б.6 и Б.7. Осуществили переход по ссылке, после чего открылось окно как представлено на изображении Б.8.

Нажали кнопку «Войти» в правой части навигационного меню, как показано на рисунке Б.10, после чего открылась форма входа, где заполнили данные, указанные в регистрации. Пример отображен на картинке Б.11. Далее нажали кнопку «Войти». В навигационном меню в правой части стало отображаться «Привет, TestLogin», «Профиль» и «Выйти», как видно на картинке Б.9.

Нажали кнопку «Выйти» в правой части навигационного меню, как показано на рисунке Б.9, после чего произошло перенаправление на главную страницу. В навигационном меню стали отображаться надписи как на рисунке Б.10.

Осуществили адресацию на форму входа, далее нажали на «Забыли пароль? Нажмите для сброса», после чего открылась форма сброса пароля, как указано на изображении Б.12. Ввели электронный адрес, далее нажали кнопку «Сбросить пароль». После перехода в почтовый адрес открыли присланное письмо с подтверждением сброса пароля на подобии указанного в примере на рисунке Б.13. Далее открыли ссылку из вложения письма, после чего открылось окно как представлено на изображении Б.14. Ввели данные, после чего нажали кнопку «Сбросить пароль».

Ввели несуществующий URL-сайта «../абвгдеж», после чего отобразилась страница, как указано на рисунке Б.15.

Выполнили вход в систему, далее нажали на «Профиль» в навигационном меню, после чего произошла адресация на представление, отображенное на иллюстрации Б.16.

В представлении профиля нажали на «Добавить чек», где отобразилась страница как на иллюстрации Б.17. Далее выбрали «Обзор», где указали расположение фотографии с чеком. После подтверждения обработки, ниже отобразилось изображение, как видно из рисунка Б.19. Далее нажали кнопку

«Обработать» после чего страница перешла в режим загрузки. После окончания произошло обновление представления.

Произвели возврат на страницу профиля, где вместо надписи: «Чеки отсутствуют», отображается таблица, видимая из картинки Б.20.

Нажали на кнопку «Редактировать профиль», после чего отобразилась страница, на которой изменили данные на требуемые, как видно из рисунка Б.18, после чего нажали кнопку «Отправить». Результат изменений представлен на иллюстрации Б.20.

В таблице выбрали чек от 2019-05-25 00:00:00, после чего вернулось представление с обработанной информацией распознанного изображения, как отображено на изображении Б.21.

Над случайно выбранными рядами продуктов нажали «Удалить строку». С каждым нажатием сумма чека пересчитывалась – результат представлен на рисунке Б.22.

Осуществили переход на страницу добавления чека, где заполнили поля в первом ряду, кроме цены. Нажали «Внести» – поле стоимости подсветилось красным, как показано на рисунке Б.23. Ввели стоимость, после чего подсветка данного поля исчезла, далее нажали на «Добавить поле +», после чего появился новый ряд, как видно из картинке Б.24.

Посередине главной странице включили видеоролик, который начал воспроизводить графический и аудио поток. Правее, в разделе «Наши решения», нажали на «Encrypted dialogue #3», после чего под ним открылся блок с информацией. Вышеописанное показано на иллюстрации Б.25.

В центральной части главной странице во втором столбце нажали на кнопку «Скачать», после чего отобразился диалог загрузки файла, как это представлено на рисунке Б.26.

Осуществили нажатие на выделенный текст «политикой конфиденциальности», который виден из изображения Б.26, после чего произошло перенаправление на соответствующую страницу, проиллюстрированную на картинке Б.27. Там же произвели нажатие на

выделенный текст «пользовательским соглашением», после чего произошло перенаправление на данное представление, отображенное на рисунке Б.28.

В четвертой колонке, содержащей заголовок «Руководство пользователя» и видимой из иллюстрации Б.26, нажали на кнопку «Перейти». Открылась страница справочной системой с отображенными заголовками и подзаголовками. По нажатию на заголовок второго уровня «Выход из учетной записи», расположенного в разделе «Авторизация», раскрылась вложенная информация, состоящая из текста и скриншота, как показано на картинке Б.29.

В «подвале» сайта нажали на «Связаться с нами» во второй колонке, после чего открылась программа для отправки писем с заполненными полями «From:» и «To:», как это видно на изображении Б.30.

В нижней части главной страницы выбрали случайную фотографию, после чего скриншот раскрылся в диалоговом режиме, как это отображено на иллюстрации Б.31.

После внесения в профиль более десяти чеков, записи перестали добавляться, а под таблицей появилась надпись: «Старые чеки», как видно на рисунке Б.32. По нажатию на вышеуказанный текст отобразилась следующая группа чеков, данная надпись пропала и проявилась следующая: «Новые чеки», как указано на картинке Б.33.

Проведенное функциональное тестирование подтвердило соответствие ожидаемых результатов с фактическими.

### **5.2.2 Полное тестирование**

В качестве полного тестирования произвели добавление чека посредством ручного ввода. Для этого удостоверили, что сервер включен и находится в рабочем состоянии, а также, что осуществлен вход в учетную запись пользователя.

Произвели переход на страницу добавления чека. Для этого на главной странице нажали в центральной части сайта в первой колонке с заголовком



«Бухгалтерский учет» на кнопку с текстом «В профиль», как видно на рисунке Б.34. Также удостоверили, что нажатие в навигационном меню, видимом на изображении Б.9, по тексту «Профиль», а также на приветствие, приводит к аналогичному результату.

Открылось представление профиля, отображенное на рисунке Б.16, где нажали на кнопку «Добавить чек». Произошла переадресация на соответствующую страницу, видимую из рисунка Б.17.

Произвели ввод текста «Что-то» в поле «Название продукта». Во время написания каждой буквы предлагался выбор сходного товара и заносился в поля: «Продукт», «Тип» и «Категория». Нажали на поле «Дата», где появился календарь, как видно из рисунка Б.35. Выставили значение от 25.05.2019.

Нажали на текстовое поле «Организация», снизу отобразился список с предложенными вариантами, где выставили значение из предложенных в «Sandbox's».

Над строкой с введенным продуктом, имеющим наименование «Что-то», нажали на текст «Удалить строку». В результате ряд полей данного продукта исчез.

Последующим действием произвели нажатие на кнопку «Добавить поле +», после чего ниже отобразился пустой ряд текстовых полей. В поле «Наименование продукта» введен текст «Хлеб Молочный», где автоматически заполнились поля данного ряда: «Продукт», «Тип» и «Категория».

Нажали на кнопку «Внести», после чего поле с наименованием осветилось по контуру красным цветом, как видно из рисунка Б.23.

Выставили фокус на поле цены, после чего нажали клавишу «в» на клавиатуре. Как следствие ничего не произошло. Произвели повторное действие с символами «@» и «G» – результат аналогичен. Далее скопировали текст из поля наименования в буфер обмена, после чего в компонент «input» стоимости произведена вставка, где также ничего не произошло. Затем была выставлена цена в «0.98».

Произвели нажатие на «Добавить поле +», после на кнопку «Внести».

Незаполненные поля осветились красным.

Во втором ряду ввели текст «Восточное вино» в «Название продукта», ценник выставили в «6.00». Поле «Продукт» заполнился как «Варенье», «Тип» как «Сладости». Третье поле второго ряда изменили на «Вино», после чего четвертое поле автоматически было выставлено в «Алкобольные напитки».

Далее произвели нажатие на кнопку «Добавить поле +», где в поле наименования ввели «Пена монтажная», стоимость выставили в «16». Остальные поля были автоматически верно выставлены.

Нажали «Внести» – страница перезагрузилась в исходное состояние, как указано на изображении Б.17.

Выполнили переход на страницу профиля, где виден добавленный чек, как представлено на иллюстрации Б.20.

В таблице чеков нажали на надпись «2019-05-25 00:00:00», после чего произошла переадресация на представление детальной информации данного чека, как видно из рисунка Б.36. Данное представление демонстрирует верность добавленной информации.

Произведенное детальное тестирование подтвердило соответствие ожидаемых результатов с фактическими, проведя полное взаимодействие с добавлением чека посредством ручного ввода.

## **6 ПРИМЕНЕНИЕ**

### **6.1 Назначение программы**

Программное средство «Система для учета и анализа личных расходов» предназначена для обработки чековых данных, чтобы в последующем использовать возможность мониторинга данных. Сюда входит возможность оперирования данными в форматах .docx и .xls после соответствующего экспортирования требуемой информации.

Система может применяться любым человеком в любом возрасте, но предпочтение отдается русскоязычному населению или использующему кириллицу, обладающим возможностью фотографировать чеки.

### **6.2 Условия применения**

К минимальным требованиям относится аппаратура, подключенная к сети Интернет с видео-дисплейным устройством, с установленной или портативной версией современного web-браузера:

- Google Chrome;
- Opera;
- Microsoft Edge;
- Safari;
- IE 9.0+;
- Mozilla Firefox.

К оптимальным требованиям относится стабильное Интернет-соединение, прилагающаяся к устройству мышь и клавиатура.

### **6.3 Справочная система**

Справочная система – это печатные руководства пользователя,

диалоговая документация и справочный текст, описывающие, как пользоваться программным продуктом.

В том случае, если у пользователя возникнут трудности при работе с программным средством, можно воспользоваться справочной системой. Для того, чтобы обратиться к разделу помощи, необходимо нажать «Справка» в подвале любого представления.

Структура справочной системы содержит следующие разделы:

- 1) раздел «Авторизация» имеет следующие подразделы:
  - а) «Регистрация» – содержит информацию о регистрации;
  - б) «Вход в учетную запись» – информирует о входе в систему;
  - в) «Выход из учетной записи» – содержит информацию о выходе из учетной записи;
  - г) «Сброс пароля» – предоставляет процесс о смене пароля.
- 2) раздел «Андроид» содержит подразделы:
  - а) «Первый запуск» – описывает основные элементы для взаимодействия, ориентирует в программе;
  - б) «Регистрация» – содержит информацию по регистрации;
  - в) «Вход в учетную запись» – информирует о входе в систему;
  - г) «Выход из учетной записи» – предоставляет данные по выходу из учетной записи;
  - д) «Загрузка чека» – описывает процесс отправления чека;
  - е) «Настройки» – содержит информацию по настройке программы;
  - ж) «Помощь» – предоставляет описание по справочной системе.

Нажатие по заголовкам и подзаголовком открывает/скрывает содержащуюся в них информацию.

Описанные процессы позволяют верно и точно воспользоваться функционалом программного средства «Система для учета и анализа личных расходов».

## **7 ОХРАНА ТРУДА**

Одной из задач охраны труда является сведение к минимальной вероятности поражения или заболевания работающего с одновременным обеспечением комфорта при максимальной производительности труда.

Обеспечение здоровых и безопасных условий труда возлагается на администрацию предприятия. Администрация обязана внедрять современные средства техники безопасности, предупреждающие производственный травматизм, и обеспечивать санитарно-гигиенические условия, предотвращающие возникновение профессиональных заболеваний работников.

Целью охраны труда является научный анализ условий труда, технологических процессов, аппаратуры и оборудования с точки зрения возможности возникновения появления опасных факторов, выделение вредных производственных веществ. На основе такого анализа определяются опасные участки производства, возможные аварийные ситуации и разрабатываются мероприятия по их устранению или ограничению последствий.

К помещениям с ПЭВМ и ВДТ предъявляются следующие требования:

- 1) не допускается размещение мест для пользователей ВДТ, ЭВМ и ПЭВМ во всех учреждениях образования в цокольных и подвальных помещениях;
- 2) площадь одного рабочего места для пользователей ВДТ, ЭВМ и ПЭВМ на базе электронно-лучевой трубки должна составлять не менее 6 кв. м;
- 3) минимальная площадь одного рабочего места для взрослых пользователей и обучающихся учреждений профессионально-технического, среднего специального и высшего образования с использованием ВДТ, ЭВМ и ПЭВМ на базе электронно-лучевой трубки может составлять не менее 4,5 кв. м при следующих условиях:
  - а) отсутствие на рабочем месте периферийных устройств

(принтер, сканер и другое);

- б) продолжительность работы должна составлять не более 4 ч в день.
- 4) площадь одного рабочего места для пользователей ВДТ, ЭВМ и ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные и другие) должна составлять не менее 4,5 кв. м;
- 5) при возведении и реконструкции зданий с помещениями для ВДТ, ЭВМ и ПЭВМ эти помещения следует проектировать высотой от пола до потолка не менее 3,0 м;
- 6) при размещении рабочих мест с ВДТ, ЭВМ и ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора) должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м;
- 7) рабочие места с ВДТ, ЭВМ и ПЭВМ в помещениях с источниками вредных производственных факторов должны размещаться в изолированных кабинах с организованным воздухообменом;
- 8) рабочие места с ВДТ, ЭВМ и ПЭВМ при выполнении творческой работы, требующей значительного умственного напряжения или высокой концентрации внимания, рекомендуется изолировать друг от друга перегородками высотой 1,5—2,0 м;
- 9) помещения, где размещаются рабочие места с ВДТ, ЭВМ и ПЭВМ, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации;
- 10) запрещается размещать рабочие места с ВДТ, ЭВМ и ПЭВМ на расстоянии менее 10 м от силовых кабелей, вводов и высоковольтных трансформаторов;
- 11) помещения, в которых для работы используются преимущественно ВДТ, ЭВМ и ПЭВМ (диспетчерские, операторские, расчетные, классы и другие), не должны граничить с помещениями, в которых

уровни шума и вибрации превышают нормируемые значения для данной категории проводимых в них работ и их типа (механические цеха, мастерские, гимнастические залы и другие) согласно Санитарным нормам и правилам, устанавливающим ПДУ шума на рабочих местах, в помещениях жилых и общественных зданий;

- 12) звукоизоляция ограждающих конструкций помещений с ВДТ, ЭВМ и ПЭВМ должна обеспечивать нормируемые параметры шума в них;
- 13) помещения с ВДТ, ЭВМ и ПЭВМ должны оборудоваться системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией;
- 14) нормируемые параметры микроклимата, ионного состава воздуха, содержание вредных веществ в нем должны отвечать требованиям Гигиенического норматива «Предельно допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами», утвержденного постановлением Министерства здравоохранения Республики Беларусь от 28 июня 2013 г. № 59;

Также в помещениях с персональными ЭВМ ежедневно должна проводиться влажная уборка.

## **8 ЭКОНОМИЧЕСКИЙ РАЗДЕЛ**

### **8.1 Определение объема и трудоемкости программного обеспечения**

В рыночных условиях программное обеспечение выступает преимущественно в виде продукции научно-технических организаций, представляющей собой функционально завершенные и имеющие товарный вид программные средства, поставляемые заказчикам и продаваемые покупателям по рыночным ценам. Все завершенные разработки ПО являются научно-технической продукцией.

Широкое применение вычислительной техники (ВТ) требует постоянного обновления и совершенствования программного обеспечения. Выбор эффективных проектов ПО требует их экономической оценки и расчета экономического эффекта. Экономический эффект у разработчика выступает в виде роста чистой прибыли (чистого дохода, ЧД или чистого дисконтированного дохода (ЧДД), научно-технической организации от реализации ПО. Экономический эффект зависит от объема затрат на разработку проекта, уровня цены на разработанные программный продукт и объема продаж.

Экономический эффект у пользователя выражается в экономии трудовых, материальных и финансовых ресурсов, которая в конечном итоге также через уровень затрат, цену и объем продаж выступает в виде роста ЧД или ЧДД пользователя.

Стоимостная оценка ПО и определение экономического эффекта у работника предполагают составление сметы затрат, которая в денежном выражении включает следующие статьи расходов:

- 1) заработную плату исполнителей, основную( $Z_o$ ) и дополнительную;
- 2) отчисления в фонд социальной защиты населения( $Z_{сз}$ );
- 3) материалы и комплектующие ( $M$ );
- 4) спецоборудование ( $P_c$ );



- 5) машинное время ( $P_m$ );
- 6) расходы на научные командировки ( $P_{нк}$ );
- 7) прочие прямые расходы ( $P_3$ );
- 8) накладные расходы ( $P_n$ );

На основе общей суммы расходов по всем статьям ( $C_p$ ) и результатов маркетинговых исследований на рынке ПО определяется плановая отпускная цена ( $C_0$ ) с учетом прибыли (рентабельности) и налогов, включаемых в цену.

Объем ПО. Базовой для расчёта плановой сметы затрат на разработку ПО является объем ПО.

Общий объем ( $V_0$ ) программного продукта определяется исходя из количества и объема функций, реализуемых программой:

$$V_0 = \sum_{i=1}^n V_i \quad (3)$$

Где  $V_i$  – объем отдельной функции ПО;  $n$  – общее число функций.

Единицы измерения объема ПО. Оценивание объема программного продукта связано с выбором наиболее подходящей единицы измерения размера продукта. В зарубежной практике получили распространение следующие единицы измерения:

- 1) количество строк в исходного кода (Lines Of Code, LOC);
- 2) функциональные точки (Function point, FP);
- 3) точки свойств (Property point, PP);
- 4) количество сущностей на диаграмме сущностей (Entity relationship diagram, ERD);
- 5) количество сущностей на диаграмме сущностей (Entity relationship diagram, DFD);
- 6) количество «квадратиков», соответствующих процессу/контролю (PSPEC/CSPEC);
- 7) количество различных элементов в составе управленческой спецификации (element);
- 8) объем документации (количество строк, quantity lines);
- 9) количество объектов, атрибутов и служб на объектной диаграмме

(subjects, attributes, services).

Несмотря на довольно значительный перечень видов единиц измерения объема ПО, наиболее широкое распространение получили лишь первые три. Причем функциональные точки и точки свойств до сих пор используются только в сочетании с количеством строк исходного кода (LOC). Все остальные виды единиц измерения применяются в основном при разработке специализированных проектов. В данном методическом пособии в качестве единиц измерения объема ПО используется строка исходного кода (LOC). Преимущества использования строк кода как единиц измерения заключаются в том, что эти единицы:

- 1) отражают сущность труда программистов;
- 2) широко распространены и могут легко адаптироваться позволяют выполнять сопоставление размеров ПО и производительности в различных группах разработчиков;
- 3) непосредственно связаны с конечным продуктом;
- 4) могут использоваться для оценки работ до завершения проекта; позволяют автоматизировать сбор данных о количестве LOC от начала до конца проекта;
- 5) дают возможность учитывать мнение разработчика об объеме ПО на основе количества написанных строк кода.

Строка исходного кода (LOC) является универсальной метрикой, так как может применяться при создании любых программных продуктов. При подсчете LOC следует придерживаться следующих рекомендаций:

- 1) учитывать «строку исходного кода» как одну, если в ней содержится лишь один оператор (если в одной строке содержатся два выполняемых оператора, разделяемых точкой с запятой, то нужно считать две строки, а если один выполняемый оператор разбит на две «физические» строки, то он будет учитываться как один оператор);
- 2) учитывать все имеющиеся выполняемые операторы,

поддерживаемые данным продуктом;

- 3) определение данных учитывать лишь один раз;
- 4) не учитывать строки, содержащие комментарии;
- 5) не учитывать отладочный код или другой временный код (пробное ПО, средства тестирования, инструменты разработки и прототипирования и другие инструментальные средства);
- 6) учитывать каждую инициализацию, вызов или включение макроса в качестве части исходного кода;
- 7) не учитывать повторно используемые операторы исходного кода.

Расчет объема программного продукта (количества строк исходного кода) предполагает определение типа программного обеспечения (Прил. 1), всестороннее техническое обоснование функций ПО и определение объема каждой функции. На стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций могут быть получены только ориентировочные (прогнозные) оценки на основе имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения действующих нормативов (Прил. 2), которые в организациях должны периодически обновляться, уточняться и утверждаться как нормативы. На основании информации о функциях разрабатываемого ПО каталогу функций определяется объем функций и общий объем ПО, который уточняется (корректируется) с учетом условий разработки ПО в организации.

Среда разработки ПО — Python, Java, JavaScript.

ПО функциональные назначения.  $V_0 = 64700$  LOC.

Таблица 10 – Перечень, объем функций программного модуля

№ функции	Наименование (содержание)	Объем функции (LOC)
1	2	3
101	Организация ввода информации	150
102	Контроль, предварительная обработка и ввод информации	450

Окончание таблицы 10 – Перечень, объем функций программного модуля

103	Анализ входного языка	660
105	Обработка входного заказа и формирование таблицы	1340
109	Организация ввода/вывода в интерактивном режиме	320
111	Управление вводом/выводом	2400
201	Генерация структуры базы данных	4200
203	Формирование баз данных	2180
204	Обработка наборов и записи базы данных	2670
207	Манипулирование данными	9550
301	Формирование последовательного файла	360
304	Обслуживание файлов	420
305	Обработка файлов	800
307	Совместная обработка группы файлов	6180
308	Управление файлами	5750
309	Формирование файла	1080
506	Обработка ошибочных и сбойных ситуаций	410
604	Справка и обучение	720
701	Математическая статистика и прогнозирование	9320
702	Расчетные задачи	14800
703	Расчет показателей	460
707	Графический вывод результатов	480
	Итого	64700

Трудоемкость разработки ПО. По общему объему ПО и нормативам затрат труда в расчете на единицу объема определяются нормативная и общая трудоемкость разработки ПО.

Нормативная трудоемкость разработки ПО. На основании принятого к расчету объема ( $V_0$ ) и категории сложности (Прил. 3) определяется нормативная трудоемкость ПО,  $T_n = 1620$  человеко-дней.

Нормативная трудоемкость ( $T_n$ ) служит основой для определения общей трудоемкости ( $T_0$ ), расчет которой осуществляется различными способами в зависимости от размера проекта.

Общая трудоемкость небольших проектов рассчитывается по форме

$$T_0 = T_n * K_{cl} * K_t * K_n \quad (4)$$

где  $K_{cl}$  - коэффициент, учитывающий сложность ПО;

$K_t$  – поправочный коэффициент, учитывающий степень использования при разработке стандартных модулей;

$K_n$  – коэффициент, учитывающий степень новизны ПО.

Категория сложности ПО. Все ПО принято подразделять на три категории сложности (Прил. 4, табл. П.4.1) в зависимости от наличия (отсутствия) следующих характеристик:

- 1) высокий уровень языкового интерфейса с пользованием;
- 2) режим работы в реальном времени;
- 3) управление удаленными объектами;
- 4) машинная графика, многомашинные комплексы;
- 5) существенное распараллеливание вычислений;
- 6) нестандартная конфигурация технических средств;
- 7) оптимизационные и особо сложные инженерные и научные расчеты;
- 8) переносимость ПО.

Влияние фактора сложности на трудоемкость учитывается умножением нормативной трудоемкости на соответствующий коэффициент сложности.

Коэффициент сложности ( $K_{cl}$ ). Посредством коэффициента сложности учитываются дополнительные затраты труда, связанные со сложностью

разрабатываемого программного продукта (Прил. 4, табл. П 4.2). Коэффициент сложности рассчитывается по формуле

$$K_C = 1 + \sum_{i=1}^n K_i \quad (5)$$

где  $K_i$  – коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;

$n$  – количество учитываемых характеристик.

$$K_C = 1 + 0,12 + 0,08 + 0,07 + 0,06 = 1,33$$

Коэффициент, учитывающий степень использования при разработке ПО стандартных модулей ( $K_T$ ). Современные технологии разработки компьютерных программ предусматривают широкое использование так называемых коробочных продуктов (пакетов, модулей, объектов), используемых для разработки заказных систем. В настоящее время уже существует обширный рынок метапрограмм многократного использования. Степень использования в разрабатываемом ПО стандартным модулям определяется их удельным весом в общем объеме проектируемого продукта (см. Прил. 4, табл., П.4.5). При определении влияния этого фактора на трудоемкость он учитывается путем умножения нормативной трудоемкости на соответствующий коэффициент.

Степень охвата реализуемых функций разрабатываемого ПО стандартным модулям, типовыми программами и ПО – до 40%. Соответственно  $K_T = 0,8$

Коэффициент новизны разрабатываемого ПО ( $K_N$ ). Сравнение характеристик разрабатываемого ПО с имеющимися аналогами позволяет определить экспертным путем степени его новизны. Если нет доступных аналогов, то ПО присваивается категория А. При установлении коэффициентов новизны учитывается степень новизны ПО и предназначение его для новых или основных типов ПК, для новых или освоенных ОС (см. Прил. 4, табл. П.4.4.). Влияние фактора новизны на трудоемкость учитывается путем умножения трудоемкости на соответствующий коэффициент новизны – 0,7

Общая трудоемкость определяется по формуле 2:

$$T_o = 1620 * 1,33 * 0,9 * 0,8 = 1551,3 \text{ чел./дн.}$$

Общая трудоемкость для крупных проектов. При решении сложных задач с длительным периодом разработки ПО трудоемкость определяется по стадиям разработки:

техническое задание (ТЗ) – исследование;  
эскизный проект (ЭП) – анализ требований;  
технический проект (ТП) – проектирование;  
рабочий проект (РП) – разработка (кодирование, тестирование);  
внедрение (ВН) – ввод в действие.

При этом на основании нормативной трудоемкости рассчитывается общая трудоемкость с учетом распределения ее по стадиям ( $T_o$ ):

$$T_o = \sum_{i=1}^n T_i \quad (6)$$

Где  $T_i$  – трудоемкость разработки ПО на  $i$ -й стадии (чел./дн.);

$n$  – количество стадий разработки.

Трудоемкость стадий определяется на основе нормативной трудоемкости с учетом сложности, новизны, степени использования в разработке стандартных модулей ПО и удельного веса трудоемкости каждой стадий в общей трудоемкости ПО:

$$T_{yi} = T_n * d_{sti} * K_c * K_t * K_n \quad (7)$$

где  $T_{yi}$  – уточнённая трудоемкость разработки ПО на  $i$ -й стадии (технического задания, эскизного проекта, технического проекта, рабочего проекта и внедрения);

$d_{sti}$  – удельный вес трудоемкости  $i$ -й стадии разработки ПО в общей трудоемкости разработки ПО;

$K_c$  – коэффициент, учитывающий сложность ПО, вводится на всех стадиях;

$K_t$  – коэффициент, учитывающий степень использования стандартных модулей ПО, вводится только на стадии рабочего проекта;

$K_n$  – коэффициент, учитывающий степень новизны ПО, вводится на всех стадиях.

Удельные веса трудоемкости стадий в общей трудоемкости ПО

определяются экспертным путем с учетом категории новизны ПО (см. Прил. 4, табл. П.4.3). При этом сумма удельных весов всех стадий в общей трудоемкости равна единице. Если стадия эскизного проекта в здании не предусмотрена, то удельный вес стадии технического проекта  $d_{тп}$  равен сумме удельных весов стадий эскизного и технического проектов ( $d_{тп}=d_{эп}+d_{тп}$ ). В этом случае, когда объединяются стадии «Технический проект» и «Рабочий проект» в одну стадию «Технорабочий проект», трудоемкость «Технорабочего проекта» определяется по формуле:

$$T_{трп}=0.85*T_{тп}+1*T_{рп} \quad (8)$$

где  $T_{трп}$  – трудоемкость стадии «Технорабочий проект»;

$K_{тп}$  – трудоемкость стадии «Технический проект»;

$K_{рп}$  – трудоемкость стадии «Рабочий проект».

Трудоемкость ПО по стадиям. Все стадии разработки ПО различаются трудоемкостью. Трудоемкость разработки стадий ПО ( $T_{уз}$ ,  $T_{уз}$ ,  $T_{уг}$ ,  $T_{ур}$ ,  $T_{ув}$ ) определяется с учетом удельного веса трудоемкости стадии в общей трудоемкости ПО ( $d$ ), сложности ( $K_c$ ), новизны ПО ( $K_n$ ) и степени использования стандартных модулей ( $K_t$ ). При этом коэффициент ( $K_t$ ) используется только на стадии «Рабочий проект» при написании исходного кода (разработки программы). Трудоемкость стадий ПО рассчитывается по следующим формулам:

$$\text{трудоемкость стадии ТЗ: } T_{уз}=T_n*K_c*d_з*K_n=1620*0,09*1,33*0,7=135,7 \quad (9)$$

$$\text{трудоемкость стадии ЭП: } T_{уз}=T_n*K_c*d_э*K_n=1620*0,07*1,33*0,7=105,6 \quad (10)$$

$$\text{трудоемкость стадии ТП: } T_{уг}=T_n*K_c*d_т*K_n=1620*0,07*1,33*0,7=105,6 \quad (11)$$

$$\text{трудоемкость стадии РП: } T_{ур}=T_n*K_c*d_p*K_n*K_t=1620*0,61*1,33*0,8*0,7=736 \quad (12)$$

$$\text{трудоемкость стадии ВН: } T_{ув}=T_n*K_c*d_в*K_n=1620*0,16*1,33*0,8=241,3 \quad (13)$$

Общая трудоемкость определяется как сумма трудоемкостей по стадиям:

$$T_y=T_{уз}+T_{уз}+T_{уг}+T_{ур}+T_{ув} \quad (14)$$

$$T_y=135,7 + 105,6 + 105,6 + 736 + 241,3 = 1324,2 \text{ человеко-дней}$$

Нормативная трудоемкость разработки ПО ( $T_n$ ) определяется согласно Прил. 3 ( $T_n$  - 119 чел./дн.), степени новизны А



Таблица 11 – Расчет общей трудоемкости разработки ПО и численности исполнителей с учетом стадий

Показатели	Стадии					Итого
	ТЗ	ЭП	ТП	РП	ВН	
1	2	3	4	5	6	7
1. Коэффициенты удельных весов трудоёмкости стадии разработки ПО (d)	0,09	0,07	0,07	0,61	0,16	1,00
2. Распределение нормативной трудоемкости ПО( $T_n$ ) по стадиям, чел./дн.	145,8	113,4	113,4	988,2	259,2	1620
3. Коэффициент сложности ПО ( $K_c$ )	1,33	1,33	1,33	1,33	1,33	
4. Коэффициент, учитывающий использование стандартных модулей				0,8		
5. Коэффициент, учитывающий новизну ПО ( $K_n$ )	0,7	0,7	0,7	0,7	0,7	
6. Общая трудоемкость ПО ( $T_y$ ), чел./дн.	135,7	105,6	105,6	736,0	241,3	1324,2

Таким образом определен объем и трудоемкость программного обеспечения, расчет численности с учетом стадий.

## 8.2 Расчёт эффективного фонда времени и численности работников

Эффективный фонд времени одного работника ( $\Phi_{эф}$ ) рассчитывается по формуле:

$$\Phi_{эф} = D_r - D_n - D_b - D_o \quad (15)$$

где  $D_r$  – количество дней в году.

$D_{п}$  – количество праздничных дней в году.

$D_{в}$  – количество выходных дней в году.

$D_o$  – количество дней отпуска.

$$\Phi_{эф} = 365 - 9 - 104 - 24 = 228 \text{ дней}$$

где Тут – уточненная трудоёмкость программного средства;

$T_{пл.}$  – рассчитаем плановую продолжительность разработки программного средства (лет).

$\Phi_{эф.}$  – эффективный фонд времени одного работника.

$$T_{пл} = 228 / 12 * 3 = 57 \text{ дней}$$

### **8.3 Расчёт основной и дополнительной заработной платы**

Основной статьей расходов на создание ПО является заработная плата работников (исполнительного) проекта, в число которых принято включать инженеров-программистов, участвующих в написании кода, руководителей проекта, системных архитекторов, дизайнеров, разрабатывающих пользовательский интерфейс, разработчиков баз данных, web-мастеров и других специалистов, необходимых для решения специальных задач в команде.

Заработная плата руководителей организации и работников вспомогательных служб (инфраструктуры) учитывается в накладных расходах.

Расчёт основной заработной платы исполнителей. Общая трудоёмкость, плановая численность работников и плановые сроки разработки ПО являются базой для расчёта основной заработной платы разработчиков проекта. Оплата труда осуществляется на основе Единой тарифной сетки Республики Беларусь (ЕСТ), в которой даны тарифные разряды и тарифные коэффициенты (Прил. 7). Действует инструкция по распределению работников внебюджетного сектора экономики Республики Беларусь по тарифным разрядам с учетом категории, должности, образования, сложности выполняемой работы и практического

опыта. Для расчёта заработной платы правительственными органами устанавливается тарифная ставка 1-го разряда. При отсутствии задолженности по платежам в бюджет и по наличию прибыли коммерческие организации имеют право повышать тариф 1-го разряда.

По данным о специфике и сложности выполняемых функций составляется штатное расписание группы специалистов-исполнителей, участвующих в разработке ПО, с определением образования, специалистов, квалификации и должности.

Месячная тарифная ставка каждого исполнителя ( $T_m$ ) определяется путем умножения действующей месячной тарифной ставки 1-го разряда ( $T_{m1}$ ) на тарифный коэффициент ( $T_k$ ), соответствующий установленному тарифному разряду:

$$T_m = T_{m1} * T_k \quad (17)$$

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленную при 40-часовой недельной норме рабочего времени расчётную среднемесячную норму рабочего времени в часах ( $\Phi_p$ ):

$$T_{\text{ч}} = T_m / \Phi_p \quad (18)$$

где  $T_{\text{ч}}$  – часовая тарифная ставка (д.е.);

$T_m$  – месячная тарифная ставка (д.е.).

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

$$Z_{oi} = \sum_{i=1}^n T_{\text{ч}i} * T_{\text{ч}} * \Phi_{\text{п}} * K \quad (19)$$

где  $n$  – количество исполнителей, занятых разработкой конкретного ПО;

$T_{\text{ч}i}$  – часовая тарифная ставка  $i$ -го исполнителя (д.е.);

$\Phi_{\text{п}}$  – плановый фонд рабочего времени  $i$ -го исполнителя (дн.);

$T_{\text{ч}}$  – количество работы в день (ч);

$K$  – коэффициент премирования.

Программист 1 категории – тарифный разряд – 14; тарифный коэффициент – 3.25;

Расчётные данные представим в виде таблицы.

Таблица 12 – Расчёт основной заработной платы

Исполнители	Тарифная ставка 1-го разряда (Тм1), тыс.р.	Тарифный коэффициент (Тк)	Тарифная ставка данного разряда (Тм). Тыс.р.	Эффективный фонд работы за месяц (Нмес)	Тарифная ставка часовая (Тч) тыс.р.	Тарифная ставка дневная (Тдн.) тыс.р.	Продолжительность участия в разработке, лн.	Коэффициент премирования Кпр	Заработная плата основная (Зо)
Программист первой категории	36,4	3,25	118	160	0,74	5,92	57	1,7	573,6

Дополнительная заработная плата на конкретное ПО ( $З_{дi}$ ) включает выплаты, предусмотренные законодательством о труде (оплата отпусков, льготных часов, времени выполнения государственных обязанностей и других выплат, не связанных с основной деятельностью исполнителей), и определяется по нормативу в процентах к основной заработной плате:

$$З_{дi} = \frac{З_{oi} * Н_d}{100} \quad (20)$$

где  $З_{дi}$  – дополнительная заработная плата исполнителей на конкретное ПО (д.е.).

$Н_d$  – норматив дополнительной заработной платы (Прил. 7)

$З_o$  – основная заработная плата в целом по организации

$$З_{дi} = \frac{573,6 * 20}{100} = 114,7 \text{ рублей}$$

Отчисления в фонд социальной защиты населения ( $З_{сзi}$ ) определяются в соответствии с действующими законодательными актами по нормативу процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу, установленному в целом по

организации:

$$З_{сзi} = \frac{(3oi+3qi)*H_{сз}}{100} \quad (21)$$

где  $H_{сз}$  – норматив отчислений в фонд социальной защиты населения (%).

$$З_{сзi} = \frac{(573,6 + 114,7) * 34}{100} = 234 \text{ рублей}$$

Отчисления в Белгосстрах ( $З_{бгс}$ ) определяются в соответствии действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей, определенной по нормативу.

$$З_{бгс} = \frac{(3oi+3qi)*H_{бгс}}{100} \quad (22)$$

$H_{бгс}$  – норматив отчислений Белгосстрах (%)

$$З_{бгс} = \frac{(573,6 + 114,7) * 0,6}{100} = 4,1 \text{ рублей}$$

Налоги, рассчитываемые от фонда оплаты труда, определяются в соответствии с действующими законодательными актами по нормативам в процентном отношении к сумме всей заработной платы, относимой на ПО.

#### **8.4 Расчёт затрат на материалы и спецоборудование**

Расходы по статье «Материалы» определяются на основании сметы затрат, разрабатываемой на ПО с учетом действующих нормативов. По статье «Материалы» отражаются расходы на магнитные бумагу, красящие ленты и другие материалы, необходимые разработки ПО. Нормы расхода для материалов в суммарном выражении ( $H_m$ ) определяются 100 в расчете на строки исходного кода (Прил. 5) или по нормативу в процентах к фонду основной заработной платы разработчиков ( $H_O$ , который устанавливается организацией (3-5%). Сумма затрат на расходные материалы рассчитывается по формуле:

$$M_i = H_m * (V_{oi}/100) \quad (23)$$

где  $H_m$  – норма расхода материалов в расчете на 100 строк исходного кода ПО (д.е.);

$V_{oi}$  – общий объем ПО (строк исходного кода) на конкретное ПО.

Или по формуле:

$$M_i = \frac{3oi \cdot H_{мз}}{100} \quad (24)$$

где  $H_{мз}$  – норма расхода материалов от основной заработной платы (%)

$$M_i = \frac{573,6 \cdot 3}{100} = 17,2 \text{ рублей}$$

Расходы по статье «Спецоборудование» ( $P_{ci}$ ) включают затраты средств на приобретение вспомогательных специального назначения технических и программных средств, необходимых для разработки конкретного ПО, включая расходы на их проектирование, изготовление, отладку, установку и эксплуатацию. Затраты по этой статье определяются в соответствии со сметой расходов, которая составляется перед разработкой ПО. Данная статья включается в смету расходов на разработку в том случае, когда приобретаются специальное оборудование или специальные программы, предназначенные для разработки и создания только данного ПО:

$$P_{ci} = \sum_{i=1}^n C_{ci} \quad (25)$$

где  $C_{ci}$  – стоимость конкретного специального оборудования (д.е.);

$n$  – количество применяемого специального оборудования.

## 8.5 Расчёт расходов на оплату машинного времени

Расходы по статье «Машинное время» ( $P_{mi}$ ) включают оплату машинного времени, которое необходимо для разработки и отладки ПО, которое определяется по нормативам (в машино-часах) на 100 строк исходного, когда ( $H_{мв}$ ) машинного времени в зависимости от характера решаемых задач и типа ПК (Прил. 6):

$$P_{mi} = C_{mi} \cdot V_{oi} / 100 \cdot H_{мв} \quad (26)$$

где  $C_{mi}$  – цена одного машино-часа (д.е.);

$V_{oi}$  – общий объём ПО (строк исходного кода);

$H_{мв}$  – норматив расхода машинного времени на отладку 100 строк

исходного кода (машино-часов).

$$P_{mi}=0,2*64700/100*12=1552,8 \text{ рублей}$$

## 8.6 Расчет прочих затрат

Расходы по статье «Прочие затраты» ( $P_{zi}$ ) на конкретное ПО включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы. Определяются по нормативу, разрабатываемому в целом по организации, в процентах к основной заработной плате:

$$P_{zi}=(Z_{oi}*N_{пз})/100 \quad (28)$$

где  $N_{пз}$  – норматив прочих затрат в целом по организации (Прил. 7)

$P_z$  – прочие затраты в целом по организации.

$$P_{zi}=(573,6*20)/100=114,7 \text{ рублей}$$

## 8.7 Составление сметы затрат

Себестоимость продукции представляет собой стоимостную оценку используемых в процессе производства продукции природных ресурсов, сырья, материалов, топлива, энергии, основных фондов, трудовых ресурсов и других затрат на ее производство и реализацию.

По элементам затрат составляются сметы затрат на производство и реализацию всей товарной продукции.

В целях систематизации данных, составим таблицу.

Таблица 13 – Смета затрат

Наименование статей затрат	Обозначение	Сумму
Основная заработная плата	$Z_{oi}$	573,6
Дополнительная заработная плата	$Z_{di}$	114,7
Отчисления в фонд социальной защиты населения	$Z_{czi}$	234

Окончание таблицы 13 – Смета затрат

Отчисления в Белгосстрах	Збгс	4,1
Материалы	M <sub>i</sub>	17,2
Машинное время	P <sub>mi</sub>	1552,8
Прочие затраты	П <sub>zi</sub>	114,7
Итого производственная себестоимость	Спр	2611,1

Кроме того, организация-разработчик осуществляет затраты на сопровождение и адаптацию программного средства. Эти затраты определяются по нормативу в процентах от производственной себестоимости по формуле:

$$P_{ci} = C_{пр} * H_{са} / 100 \quad (30)$$

Спр – производственная себестоимость;

H<sub>са</sub> – норматив отчисления на сопровождение и адаптацию программного средства (Прил. 7).

$$P_{ci} = 2611,1 * 20 / 100 = 522,2 \text{ рублей}$$

Таким образом полная себестоимость разработки программного средства определяется по формуле:

$$C_{пол} = C_{пр} + P_{ci} \quad (31)$$

$$C_{пол} = 2611,1 + 522,2 = 3133,3 \text{ рублей}$$

## 8.8 Расчёт прогнозируемой отпускной цены

Цена — это денежное выражение стоимости единицы товара. Отпускная цена производителя — это цена, по которой производитель реализует продукцию оптово-сбытовым организациям. Она включает издержки производства и реализации прибыль, налог на добавленную стоимость. Прогнозируемая отпускная цена рассчитывается по формуле:

$$Ц = C_{пол} + П + НДС \quad (32)$$

Рентабельность и прибыль по создаваемому ПО (П) определяются, исходя из результатов анализа рыночных условий, переговоров с заказчиком



(потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость.

В случае разработки ПО для использования внутри организации оценка программного продукта производится по действующим правилам и показателям внутреннего хозрасчета (по ценам, устанавливаемым для расчета за услуги между подразделениями), на основании действующего распорядка. Прибыль рассчитывается по формуле:

$$П = C_{\text{пол}} * U_p / 100 \quad (33)$$

где П – прибыль от реализации ПО заказчику (д.е.);

$U_p$  – уровень рентабельности ПО (%); (Прил. 7)

$C_{\text{пол}}$  – полная себестоимость ПО (д.е.).

$$П = 3133,3 * 30 / 100 = 940 \text{ рублей}$$

В соответствии с действующим законодательством в цену программного средства включается налог на добавленную стоимость. Налог на добавленную стоимость рассчитывается по формуле:

$$\text{НДС} = (C_{\text{пол}} + П) * C_{\text{ндс}} / 100 \quad (34)$$

где  $C_{\text{пол}}$  – полная себестоимость разработки программного средства за весь период;

П – прибыль

$C_{\text{ндс}}$  – ставка налога на добавленную стоимость (по действующему законодательству).

$$\text{НДС} = (3133,3 + 940) * 20 / 100 = 814,7 \text{ рублей}$$

Результат расчётов заносится в таблицу.

Таблица 14 – Прогнозируемая отпускная цена

Наименование статей затрат	Обозначение	Сумма
Полная производственная себестоимость	Спол	3133,3
Прибыль и рентабельность по создаваемому программному средству	П	940

Окончание таблицы 14 – Прогнозируемая отпускная цена

Налог на добавленную	НДС	814,7
Прогнозируемая отпускная цена	Ц	4888,0

Вышеописанная таблица демонстрирует произведенные расчеты.

## 8.9 Расчёт чистой прибыли и прибыли от реализации программного средства

Прибыль от реализации продукции (Прп) рассчитывается как выручка от реализации товаров (работ, услуг) за минусом налогов, включаемых в цену продукции выплачиваемых из выручки, себестоимости реализованных товаров (работ, слуг), а также расходов на реализацию (если они не включены в себестоимость). Расчет производится по формуле:

$$П_{рп} = V_p - C - НДС \quad (35)$$

где  $V_p$  – выручка от реализации продукции,  $C$  – затраты на производство и реализацию, НДС – налог на добавленную стоимость

Налог на добавленную стоимость рассчитывается по формуле:

$$НДС = V_p * C_{ндс} / (100 + C_{ндс}) \quad (36)$$

где  $V_p$  – выручка от реализации продукции;

$C_{ндс}$  – ставка налога на добавленную стоимость (по действующему законодательству).

$$НДС = 4888,0 * 20 / (100 + 20) = 814,7 \text{ рублей}$$

$$П_{рп} = 4888,0 - 3133,3 - 814,7 = 940 \text{ рублей}$$

Чистая прибыль (ЧП) рассчитывается как разница налогооблагаемой прибыли и суммы налога на прибыль по формуле:

$$ЧП = П_{рп} - Н_{нпр} \quad (37)$$

где  $П_{рп}$  – прибыль от реализации,  $C_{нпр}$  – ставка налога на прибыль (в соответствии с действующим законодательством).

$$\text{Налог на прибыль} = 940 * 18 / 100 = 169,2 \text{ рублей}$$

$$\text{ЧП} = 940 - 169,2 = 770,8 \text{ рублей}$$

## 8.10 Расчёт показателей эффективности от внедрения программного средства

К показателям эффективности от внедрения программного средства относятся:

- 1) срок окупаемости проекта ( $T_{\text{ок}}$ );
- 2) коэффициент эффективности программного обеспечения;
- 3) рентабельность затрат (себестоимость).

Срок окупаемости рассчитывается по формуле:

$$T_{\text{ок}} = 3 / \text{ЧП} \quad (38)$$

где 3 – затраты, связанные с разработкой и реализацией программного обеспечения (полная себестоимость);

ЧП – чистая прибыль.

$$T_{\text{ок}} = 3133,3 / 770,8 = 4,1 \text{ месяцев}$$

Коэффициент эффективности программного обеспечения ( $K_{\text{эф}}$ ) рассчитывается по формуле:

$$K_{\text{эф}} = 1 / T_{\text{ок}} \quad (39)$$

где  $T_{\text{ок}}$  – срок окупаемости проекта

$$K_{\text{эф}} = 1 / 4,1 = 0,24$$

Рентабельность затрат рассчитывается по формуле:

$$P_z = \text{ЧП} / 3 * 100 \quad (40)$$

где ЧП – чистая прибыль;

3 – затраты, связанные с разработкой и реализацией программного обеспечения (полная производственная себестоимость).

$$P_z = 770,8 / 3133,3 * 100 = 24,6\%$$

В итоге были произведены расчеты для определения общей стоимости программного продукта, основной и дополнительной заработной платы, стоимостные затраты на материалы и прочие потребности.

## ЗАКЛЮЧЕНИЕ

Задачей дипломного проекта являлась разработка программного средства «Система для учета и анализа личных расходов», состоящего из web-сервера, предоставляющего: rest full api, управляемые формы, в том числе отображения страниц на основе паттерна MVC; и взаимодействующего с android- и web-клиентом, представляя собой комплекс взаимосвязанного программного обеспечения по восприятию визуальных образов и самостоятельно занесенных данных чека с их дальнейшим преобразованием и предоставлением анализа на основе выборке данной информации в составлении учета личных затрат потребителя.

В ходе написания программного решения была разработана функционирующая система с использованием языков программирования «Java», «Python», «JavaScript», «JQuery» и «Ajax». Также с помощью языка структурированных запросов «SQL», языка гипертекстовой разметки и каскадной таблицы стилей.

Была реализована база данных на основе подхода «CodeFirst» посредством «SQL-Alchemy» и внутренней библиотеки «flask-alchemy», на основе системы управления базами данных «PostgreSQL». Также осуществлены все функции программы и api. Результатом выше сказанного образован готовый программный продукт.

Основное внимание уделено изучению способов проектирования приложений, объектно-ориентированному и системному программированию.

Уникально разработанная справочная система позволяет в совершенстве овладеть всеми пользовательскими функциями программного средства, исключая двусмысленность и неточность подачи информации, тем самым позволяя пользователям реализовать максимальный потенциал и упростить взаимодействие с данным проектом. Отображение в структуре web-клиента позволяет использовать справочные данные на любом устройстве.

Были проведены испытания программы на правильность получаемых

результатов. Выявленные ошибки устранены в ходе данных работ. Тестирование производилось на основании выбора различных методологий, использованных в случайном порядке. Результатом проведенного тестирования установлено, что программа удовлетворяет поставленным перед ней требованиям и является готовым программным средством.

Основные модификации по дальнейшему улучшению заключаются в:

- 1) реинжиниринге;
- 2) интернационализации и локализации;
- 3) создании собственной библиотеке оптического распознавания символов или улучшения существующей open-source библиотеки;
- 4) расширении списка экспорта отчетной документации;
- 5) улучшении и расширении подачи информации на основе анализа данных затрат;
- 6) добавлении взаимодействия с графами доходов;
- 7) портировании и миграции программного обеспечения.

Достоинством проекта является простота и интуитивность программного средства, упрощение отслеживания личных затрат в повседневной жизни человека.

Исходя из вышесказанного можно утверждать о выполненной цели дипломного проекта, реализовавшего поставленные задачи в соответствии с заданным условием.

## ЛИТЕРАТУРА

1. Бейзер, Б. Тестирование черного ящика: технологии функционального тестирования программного обеспечения и систем / Б. Бейзер – Питер, 2004.
2. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. – М.: СПб: Символ-Плюс, 2014. – 560 с.
3. Шаймарданов, Р.Б. Моделирование и автоматизация проектирования структур баз данных / Р.Б. Шаймарданов. – М.: Радио и связь, 2013. – 120 с.
4. Малыхина, М. Базы данных: основы, проектирование, использование / М. Малыхина. – М.: БХВ-Петербург, 2015. – 512 с.
5. Хайкин, С. Нейронные сети. Полный курс / С. Хайкин – Вильямс, 2018.
6. Каллан, Р. Основные концепции нейронных сетей / Р. Каллан – Вильямс, 2003.
7. Куликов, С. Тестирование программного обеспечения. Базовый курс / С. Куликов – Четыре четверти, 2017
8. Мэтиз, Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. – СПб.: Питер, 2017. — 496 с.
9. Доусон, М. Програмируем на Python / М. Доусон. – СПб.: Питер, 2014. – 416с.
10. Любанович, Б. Простой Python. Современный стиль программирования / Б. Любанович. – Питер, 2017. – 480с.
11. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. – Символ, 2015. – 608с.
12. Лутц, М. Python. Карманный справочник /М. Лутц. – Вильямс, 2015. – 320с.
13. Гринберг, М. Разработка веб-приложений с использованием Flask

на языке Python / М. Гринберг – ДМК пресс, 2014. – 282с.

14. Гибсон, У. Распознавание образов / У. Гибсон – Издательство Азбука, 2015. – 384с.

15. Макфарланд, Д. Новая большая книга css / Д. Макфарланд – Питер, 2019. – 720с.

16. Макфарланд, Д. JavaScript и JQuery. Исчерпывающее руководство / Д. Макфарланд – Эксмо, 2016. – 880с.

17. Бэер, Б. JQuery. Подробное руководство по продвинутому JavaScript / Б. Бэер – Символ, 2012. – 624с.

18. Пауэрс, Ш. Добавляем Ajax / Ш. Пауэрс – O’relly, 2009. – 448с.

19. Бенкен, Е. Ajax программирование для Интернета / Е. Бенкен – Питер, 2009. – 464с.

20. Голощапов, А. Google Android программирование для мобильных устройств / Голощапов А.Л. – Санкт-Петербург, 2011.

21. Хашими, С. Разработка приложений для Android / Хашими С., Коматинени С., Маклин Д. – Эксмо, 2011.

22. Дейтел, П. Android для программистов/ Х. Дейтел, Э. Дейтел, М. Моргано – Питер, 2013.

23. Майер, Р. Android 2. Программирование приложений для планшетных компьютеров и смартфонов/ Майер Р. – Эксмо, 2013.

24. Бурнет, Э.Привет, Android! Разработка мобильных приложений/ Э. Бурнет – СПб Питер, 2012.

25. Герберт, Ш. Java 8. Полное руководство; 9-е изд/ Ш. Герберт – Вильяме, 2015.

26. Дэрсси, Л. Android за 24 часа. Программирование приложений под операционную систему Google/ Л. Дэрсси, К. Шейн – Рид Групп, 2012.

27. Колиснеченко, Д. Программирование для Android. Самоучитель / Д. Колиснеченко – Санкт-Петербург, 2011.

28. Блэйк, М. Программирование под Android / М. Блэйк – Санкт-Петербург, 2012.

## ПРИЛОЖЕНИЕ А

(обязательное)

### Текст программы

#### Модуль app.py:

```
from flask import send_from_directory
from session import *
from flask_bootstrap import Bootstrap
from project.controllers import user_controller, authorization_controller, home_controller, support_controller
from project.controllers.api.v1 import user_controller, organization_controller, product_controller, check_controller
from project.util import errors
data_check = None
@app.route('/favicon.ico')
def favicon():
    return send_from_directory(os.path.join(app.root_path, 'static'),
                              'favicon.ico', mimetype='image/vnd.microsoft.icon')
if __name__ == '__main__':
    app.config.from_object('config.DevelopmentConfig')
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
    app.config.from_object('config')
    csrf.init_app(app)
    api.WTF_CSRF_ENABLED = False
    bootstrap = Bootstrap(app)
    Bootstrap(app)
    app.run(host='0.0.0.0', port='5000')
```

#### Модуль project/controllers/api/v1/user\_controller.py:

```
import base64
import datetime
import uuid
from flask import jsonify, request
from project.models import user
from project.models.abstract_product import AbstractProduct
from project.models.check import Check
from project.models.organization import Organization
from project.models.product import Product
from project.models.user import User
from project.util.check_process.check_parser import highlight_abstract_product, DataCheck, check_parser,
product_parser
from project.util.check_process.check_processing import start_check_process
from project.util.email import send_confirm_email
from project.util.image_process.image_processing import start_abbyy
from session import *
BASE_URL = '/api/v1/user/'
@app.route(BASE_URL + 'get_user', methods=["POST"])
@csrf.exempt
def get_user():
    dict_body = request.get_json()
    user_ = User.query.filter_by(public_id=dict_body['public_id']).first()
    return jsonify({"public_id": user_.public_id, "email": user\_email, "password": user_.password_hash, "username":
    user_.username, "registered_on": str(user_.registered_on), "confirmed": user_.confirmed, "wage": str(user_.wage),
    "confirmed_on": str(user_.confirmed_on)})
@app.route(BASE_URL + 'create_user', methods=["POST"])
@csrf.exempt
def create_user():
```



```

dict_body = request.get_json()
user_ = user.User.query.filter_by(username=dict_body['username']).first()
if user_:
    return jsonify({'status': 'fail', 'message': 'Username already exists.'}), 409
user_ = user.User.query.filter_by(email=dict_body['email']).first()
if not user_:
    data = user.User(
        public_id=str(uuid.uuid4()),
        email=dict_body['email'],
        username=dict_body['username'],
        password=dict_body['password'],
        registered_on=datetime.datetime.utcnow(),
        confirmed=False,
        wage=dict_body['wage'],
    )
    manual_session.add(data)
    manual_session.commit()
    send_confirm_email(data)
    return jsonify({'status': 'success', 'message': 'New user successfully created.'}), 200
else:
    return jsonify({'status': 'fail', 'message': 'User already exists. Please Log in.'}), 409
@app.route(BASE_URL + 'check_user', methods=["POST"])
@csrf.exempt
def check_user():
    dict_body = request.get_json()
    user_ = user.User.query.filter_by(email=dict_body['email']).first()
    if user_ is None or not user_.check_password(dict_body['password']) or not user_.confirmed:
        pass
    else:
        return jsonify({'public_id': user_.public_id, 'email': user\_email}), 200
    return jsonify({'status': 'fail', 'message': 'Invalid username or password. Or not confirmed e-mail.'}), 409
@app.route(BASE_URL + '/upload_image', methods=["POST"])
@csrf.exempt
def upload_image():
    dict_body = request.get_json() # convert body to dictionary
    user_public_id = dict_body['public_id']
    img_data = base64.b64decode(dict_body['b64_jpg'])
    # I assume you have a way of picking unique filenames
    dynamic_name = str(uuid.uuid4())
    file_name = DIRECTORY_TEMP_FILES + dynamic_name
    source_file = file_name + '.jpg'
    target_file = file_name + '.xml'
    with open(source_file, 'wb') as f:
        f.write(img_data)
    start_abbyy(target_file, source_file)
    start_check_process(target_file, user_public_id)
    os.remove(source_file)
    os.remove(target_file)
    return jsonify({'message': 'Someone get pack.'}), 200

```

### Модуль project/controllers/api/v1/organization\_controller.py:

```

from flask import jsonify, request
from session import *
from project.models import organization
BASE_URL = '/api/v1/organization'
@app.route(BASE_URL + '/organizations', methods=['GET'])
def get_organizations():
    organizations = organization.Organization.get_all()
    results = []
    for organization_ in organizations:
        obj = {

```

```

'id': organization_id,
'legal_name': organization.legal_name,
'legal_address': organization.legal_address,
'taxpayer_identification_number': organization.taxpayer_identification_number,
}
results.append(obj)
response = jsonify(results)
response.status_code = 200
return response
@app.route(BASE_URL + '/create_organization', methods=["POST"])
@csrf.exempt
def create_organization():
dict_body = request.get_json() # convert body to dictionary
organize = organization.Organization(legal_name=dict_body['legal_name'],
legal_address=dict_body['legal_address'],
taxpayer_identification_number=dict_body['taxpayer_identification_number'],
)
manual_session.add(organize)
manual_session.commit()
return jsonify({'message': 'New organization successfully created.'}), 200

```

### Модуль project/controllers/authorization\_controller.py:

```

import datetime
import uuid
from flask import redirect, url_for
from flask_login import login_user, logout_user
from werkzeug.urls import url_parse
from project.controllers.user_controller import *
from project.forms import RegistrationForm, LoginForm, ResetPasswordRequestForm, ResetPasswordForm
from project.util.email import send_password_reset_email, send_confirm_email
BASE_URL = '/authorization/'
@app.route(BASE_URL + 'login', methods=['GET', 'POST'])
def login():
if current_user.is_authenticated:
return redirect(url_for('index'))
form = LoginForm()
if form.validate_on_submit():
user_ = User.query.filter_by(username=form.username.data).first()
if user_ is None or not user_.check_password(form.password.data) or not user_.confirmed:
flash('Неверный логин или пароль. Или учетная запись не подтверждена.')
return redirect(url_for('login'))
login_user(user_, remember=form.remember_me.data)
next_page = request.args.get('next')
if not next_page or url_parse(next_page).netloc != "":
next_page = url_for('index')
return redirect(next_page)
return render_template('authorization/login.html', title='Вход', form=form)
@app.route(BASE_URL + 'register', methods=['GET', 'POST'])
def register():
if current_user.is_authenticated:
return redirect(url_for('index'))
form = RegistrationForm()
if User.query.filter_by(email=form.email.data).first():
flash('E-mail уже используется')
return render_template('authorization/register.html', title='Регистрация', form=form)
if User.query.filter_by(username=form.username.data).first():
flash('Логин уже используется')
return render_template('authorization/register.html', title='Регистрация', form=form)
if form.validate_on_submit():
user_ = User(username=form.username.data,
email=form.email.data,

```

```

confirmed=False,
public_id=str(uuid.uuid4()),
registered_on=datetime.datetime.utcnow(),
password=form.password.data
db.session.add(user_)
db.session.commit()
send_confirm_email(user_)
flash('Вам было отправлено письмо с подтверждением создания аккаунта.', 'success')
return redirect(url_for('login'))
return render_template('authorization/register.html', title='Регистрация', form=form)
@app.route(BASE_URL + 'logout')
def logout():
    logout_user()
    return redirect(url_for('index'))
@app.route(BASE_URL + 'reset_password_request', methods=['GET', 'POST'])
def reset_password_request():
    if current_user.is_authenticated:
        return redirect(url_for('index'))
    form = ResetPasswordRequestForm()
    if form.validate_on_submit():
        user_ = User.query.filter_by(email=form.email.data).first()
        if user_:
            send_password_reset_email(user_)
            flash('Check your email for the instructions to reset your password')
            return redirect(url_for('login'))
        return render_template('authorization/reset_password_request.html', title='Сброс пароля', form=form)
    @app.route(BASE_URL + 'reset_password/<token>', methods=['GET', 'POST'])
    def reset_password(token):
        if current_user.is_authenticated:
            return redirect(url_for('index'))
        user_ = User.verify_reset_password_token(token)
        if not user_:
            return redirect(url_for('index'))
        form = ResetPasswordForm()
        if form.validate_on_submit():
            user_.password = form.password.data
            db.session.commit()
            flash('Ваш пароль был сброшен.')
            return redirect(url_for('login'))
        return render_template('authorization/reset_password.html', form=form)
    @app.route(BASE_URL + 'confirm_email/<token>', methods=['GET', 'POST'])
    def confirm_email(token):
        if current_user.is_authenticated:
            return redirect(url_for('index'))
        user_ = User.verify_confirm_email_token(token)
        if not user_:
            return redirect(url_for('index'))
        user_.confirmed = True
        user_.confirmed_on = datetime.datetime.utcnow()
        db.session.commit()
        return render_template('authorization/confirm_email.html')

```

### Модуль project/util/check\_process/check\_parser.py:

```

import re
import dateutil.parser as date_parser
from datetime import datetime
from project.models import organization, check, product
class DataCheck:
    Date = "
    Unp = "
    LegalName = "

```

```

LegalAddress = "
Products = { }
ResultPrice = 0
AbstractProducts = []
def check_parser(line, data_check):
    upper_text = line.upper()
    if data_check.LegalName == "":
        data_check.LegalName, line = legal_name_parse(line)
    if data_check.LegalAddress == "":
        data_check.LegalAddress, line = legal_address_parse(line)
    if data_check.Unp == "":
        data_check.Unp, line = unp_parse(upper_text, line)
    if data_check.Date == "":
        data_check.Date, line = date_parse(upper_text, line)
    if line.find('ИТОГ') != -1 or line.find('К ОПЛАТЕ') != -1:
        result_price = re.sub(r's+', ' ', line)
        match = re.search(r'\d+[\.\,]\d{2}',
            result_price[0:len(result_price) - 1])
        try:
            if match is not None:
                data_str = match.group(1)
                data_check.ResultPrice = data_str
            except ValueError as err:
                print(err)
        if upper_text.find('МАРШРУТ') != -1 or upper_text.find('ЭКСПРЕСС') != -1:
            data_check.Products['Маршрут'] = 0
        return remove_rudiments(line)
    def product_parser(line, previous_line, data_check):
        price = product_price_parse(line)
        if price != line:
            price.replace(',', '.')
        product_name = product_price_parse(previous_line)
        if product_name != previous_line:
            product_name = line[:len(line)-len(price)]
            product_name.replace("\n", "")
            if product_name.upper().find('ДИСКОНТ') != -1 or product_name.upper().find('СКИДКА') != -1\
            or product_name.upper().find('ОПЛАЧЕНО') or product_name.upper().find('СДАЧА')\
            or product_name.upper().find('СЕРТИФИКАТ'):
                return line
            product_name = product_name.strip()
            data_check.Products[product_name] = price
        return line
    def highlight_abstract_product(product_name):
        product_name = ".join([i for i in product_name if not i.isdigit()])
        product_name = product_name.strip()
        abstract_product = product_name[:product_name.find(' ')]
        return abstract_product
    def product_price_parse(line):
        result_price = re.sub(r's+', ' ', line[:-1])
        match = re.search(r'\d{2}[\.\, ]\d+',
            result_price[:len(result_price) - 1])
        try:
            if match is not None:
                price_str = match.group(1)
                for ch in [',', '\', ' ']:
                    if ch in price_str:
                        price_str = price_str.replace(ch, '.')
                return price_str[:-1]
            except ValueError as err:
                print(err)
        return line
    def date_parse(upper_text, line):

```

```

if upper_text.find("ДАТА") != -1:
    data_str = upper_text[upper_text.rfind("ДАТА") + 4:upper_text.rfind("ДАТА") + 15]
if upper_text.find("ВРЕМЯ") != -1:
    data_str += upper_text[upper_text.rfind("ВРЕМЯ") + 5:upper_text.rfind("ВРЕМЯ") + 12]
else:
    data_str = ".join(
    i for i in upper_text if
    (i.isdigit() or i == '/' or i == '\' or i == ':' or i == '.' or i == ' ' or i == '-'))
    data_str = re.sub(r'\s+', ' ', data_str)
    match = re.search(r'(\d{2}[/.:-]\d{2}[/.:-]\d{2,4}([ ——]\d{2}[:]\d{2}(\d{2})*))',
    data_str[0:len(data_str) - 1])
    try:
    if match is not None:
    data_str = match.group\(1\)
    else:
    return ", line
    except ValueError:
    return ", line
    try:
    data_str = date_parser.parse(data_str, fuzzy=True, dayfirst=True)
    if data_str.year < 2000:
    return ", line
    except ValueError:
    return ", line
    return data_str, "
    def unp_parse(upper_text, line):
    unp_str = "
    if upper_text.find("УИП") != -1:
    unp_str = upper_text[upper_text.rfind("УИП") + 3:upper_text.rfind("УИП") + 13]
    unp_str = ".join(i for i in unp_str if (i.isdigit()))
    if unp_str == "":
    return ", line
    return unp_str, "
    def legal_name_parse(line):
    legal_name_str = "
    legal_abbreviation = ['ЧТУП', 'ООО', 'ОАО', 'ЗАО', 'ОДО', 'ЧУП', ]
    for abbreviation in legal_abbreviation:
    legal_name_str = search_organization(abbreviation, line)
    if legal_name_str != "":
    try:
    int(legal_name_str[4:7])
    continue
    except ValueError:
    pass
    legal_name_str = legal_name_str[0:legal_name_str.find('\n')]
    break
    if legal_name_str == "":
    return ", line
    return legal_name_str.strip(), "
    def search_organization(abbreviation, text):
    if text.find(abbreviation) != -1:
    return text[text.find(abbreviation)]
    return "
    def legal_address_parse(line):
    upper_text = line.upper()
    legal_address_str = "
    if upper_text.find("ИП.") != -1 or upper_text.find("УЛ.") != -1 or upper_text.find("Г.") != -1 \
    or upper_text.find("ИП-Т") != -1 or upper_text.find("ИП-КТ.") != -1 or upper_text.find("Г.") != -1:
    legal_address_str = None
    if legal_address_str is not None:
    return ", line
    return line.strip(), "

```

```

def remove_rudiments(line):
if line.find('ЗБД') != -1 or line.find('CKHO') != -1 or line.find('ПЕГ') != -1 or line.find('КАЦКА') != -1 or \
line.find('ЧЕК') != -1 or line.find('Кассир') != -1 or line.find('П/Н') != -1 or line.find('ККМ') != -1 \
or line.find('ИИИ') != -1 or line.find('ДЮК') != -1 or line.find('ККТ') != -1 or line.find('ПРИХОД') != -1 \
or line.find('ФН') != -1 or line.find('ЭКЗ') != -1 or line.find('ФП') != -1 or line.find('ФД') != -1 \
or line.find('ЗН') != -1:
return ""
return line

```

## Модуль config.py:

```

import os
from flask import Flask, Blueprint
from flask_babel import Babel
from flask_bootstrap import Bootstrap
from flask_login import LoginManager
from flask_mail import Mail
from flask_sqlalchemy import SQLAlchemy
from flask_wtf import CSRFProtect
from flask_restful import Api
from constants import *
app = Flask(__name__)
api = Api(app)
app.config['SQLALCHEMY_DATABASE_URI'] = DB_STRING
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
app.config.update(dict(
    SECRET_KEY=SECRET_KEY,
    WTF_CSRF_SECRET_KEY=WTF_CSRF_SECRET_KEY
))
WTF_CSRF_ENABLED = True
csrf = CSRFProtect(app)
api.WTF_CSRF_ENABLED = False
api = Api(app, decorators=[csrf.exempt])
api_blueprint = Blueprint('api', __name__)
csrf.exempt(api_blueprint)
app.register_blueprint(api_blueprint)
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])
UPLOAD_FOLDER = '/home/delictumest/Downloads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
mail_settings = {
    "MAIL_SERVER": MAIL_SERVER,
    "MAIL_PORT": MAIL_PORT,
    "MAIL_USE_TLS": MAIL_USE_TLS,
    "MAIL_USE_SSL": MAIL_USE_SSL,
    "MAIL_USERNAME": MAIL_USERNAME,
    "MAIL_PASSWORD": MAIL_PASSWORD
}
app.config.update(mail_settings)
mail = Mail(app)
bootstrap = Bootstrap(app)
app.config['LANGUAGES'] = LANGUAGES
# app.config.from_pyfile('babel.cfg')
babel = Babel(app)
app.config['OAUTH_CREDENTIALS'] = {
    'facebook': {
        'id': '307880843191110',
        'secret': 'b9241e42ccde9e6f5cbc777d9d1dee83'
    },
    'twitter': {
        'id': '3RzWQclolxWZIMq5LJqzRZPTI',
        'secret': 'm9TEd58DSEtRrZHpz2EjrV9AhsBRxKMo8m3kuIZj3zLwzwlimt'
    }
}

```

```

    }
}
login = LoginManager(app)
login.login_view = 'login'
basedir = os.path.abspath(os.path.dirname(__file__))
class Config:
    SECRET_KEY = os.getenv('SECRET_KEY', 'my_precious_secret_key')
    DEBUG = False
class DevelopmentConfig(Config):
    # uncomment the line below to use postgres
    # SQLALCHEMY_DATABASE_URI = postgres_local_base
    DEBUG = True
    SQLALCHEMY_DATABASE_URI = DB_STRING
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    ASSETS_DEBUG = True
class TestingConfig(Config):
    DEBUG = True
    TESTING = True
    PRESERVE_CONTEXT_ON_EXCEPTION = False
    SQLALCHEMY_TRACK_MODIFICATIONS = False
class ProductionConfig(Config):
    DEBUG = False
config_by_name = dict(
    dev=DevelopmentConfig,
    test=TestingConfig,
    prod=ProductionConfig
)
key = Config.SECRET_KEY

```

### Модуль session.py:

```

from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from config import *
def create_session(config):
    engine = create_engine(config['SQLALCHEMY_DATABASE_URI'])
    Session = sessionmaker(bind=engine)
    session = Session()
    session._model_changes = {}
    return session
manual_session = create_session(app.config)

```

### Модуль project/controllers/api/v1/check\_controller.py:

```

from datetime import datetime
from flask import jsonify, request

from project.models.check import Check
from project.models.product import Product
from session import *
from config import db
BASE_URL = '/api/v1/check'
@app.route(BASE_URL + '/change_date', methods=['POST'])
@csrf.exempt
def change_date():
    check = Check.get_by_id(request.json['check_id'])
    check.date_time_of_purchase = datetime.strptime(request.json['date_time'], '%Y-%m-%d')
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200

@app.route(BASE_URL + '/remove_check', methods=['POST'])
@csrf.exempt

```

```

def remove_check():
    check = Check.get_by_id(request.json['check_id'])
    products = Product.get_all_check_products(check.id)
    for product in products:
        db.session.delete(product)
        db.session.commit()
    db.session.delete(check)
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/total_cost/<id>', methods=['GET'])
def total_cost(id):
    id = int(id)
    if id < 0:
        return jsonify({'message': 'Fail.'}), 404
    products = Product.query.filter_by(check_id=id).all()
    total_cost = 0
    for product in products:
        total_cost += product.product_price
    results = [str(total_cost)]
    response = jsonify(results)
    response.status_code = 200
    return response

```

### Модуль project/controllers/api/v1/product\_controller.py:

```

from flask import jsonify, request
from project.models.abstract_product import AbstractProduct
from project.models.category_type import CategoryType
from project.models.product import Product
from project.models.product_category import ProductCategory
from session import *
from project.models import organization
BASE_URL = '/api/v1/product'
@app.route(BASE_URL + '/get_all_product_fields_info_by_id/<product_id>', methods=['GET'])
def get_all_product_fields_info_by_id(product_id):
    if int(product_id) < 0:
        return jsonify({'message': 'Fail.'}), 404
    db_product = Product.get_product_by_id(int(product_id))
    db_abstract_product = AbstractProduct.get_by_id(db_product.abstract_product_id)
    product_category = ProductCategory.get_by_id(db_abstract_product.product_category_id)
    category_type = CategoryType.get_by_id(product_category.category_type_id)
    results = [db_product.product_name, str(db_product.product_price), db_abstract_product.product_name,
               product_category.product_category_name, category_type.category_type_name]
    response = jsonify(results)
    response.status_code = 200
    return response
@app.route(BASE_URL + '/get_all_product_info/<product_name>', methods=['GET'])
def get_all_product_info(product_name):
    if len(product_name) < 2:
        return jsonify({'message': 'Fail.'}), 404
    abstract_product = product_name.capitalize()
    db_abstract_product = AbstractProduct.get_product(abstract_product)
    if db_abstract_product is None:
        temp_str = abstract_product[0]
        i = 1
        while len(abstract_product) > i and abstract_product[i].islower():
            abstract_product = abstract_product[i]
            i += 1
        db_abstract_product = AbstractProduct.get_product(temp_str)
    if db_abstract_product is None:
        temp_ap = abstract_product[1:abstract_product.find(' ')]
        abstract_product = abstract_product[0:abstract_product.find(' ')]

```



```

    db_abstract_product = AbstractProduct.get_product(abstract_product)
if db_abstract_product is None:
    part_len = int(len(abstract_product) / 2)
    db_abstract_product = AbstractProduct.get_product(abstract_product[:part_len])
    if db_abstract_product is None:
        db_abstract_product = AbstractProduct.get_product('Неопределено')
    product_category = ProductCategory.get_by_id(db_abstract_product.product_category_id)
    category_type = CategoryType.get_by_id(product_category.category_type_id)
    results = [db_abstract_product.product_name, product_category.product_category_name,
category_type.category_type_name]
    response = jsonify(results)
    response.status_code = 200
    return response
@app.route(BASE_URL + '/get_category_type/<product_category>', methods=['GET'])
def get_category_type(product_category):
    if len(product_category) < 3:
        return jsonify({'message': 'Fail.'}), 404
    product_category = product_category.capitalize()
    db_product_category = ProductCategory.get_by_name(product_category)
    if db_product_category is None:
        temp_str = product_category[0]
        i = 1
        while len(product_category) > i and product_category[i].islower():
            temp_str += product_category[i]
            i += 1
        db_product_category = ProductCategory.get_by_name(temp_str)
    if db_product_category is None:
        abstract_product = product_category[0:product_category.find(' ')]
        db_product_category = ProductCategory.get_by_name(abstract_product)
    if db_product_category is None:
        part_len = int(len(product_category) / 2)
        db_product_category = ProductCategory.get_by_name(product_category[:part_len])
        if db_product_category is None:
            db_product_category = ProductCategory.get_by_name('Неопределено')
        category_type = CategoryType.get_by_id(db_product_category.category_type_id)
        results = [db_product_category.product_category_name, category_type.category_type_name]
        response = jsonify(results)
        response.status_code = 200
        return response
@app.route(BASE_URL + '/get_all_product_info_by_id/<id>', methods=['GET'])
def get_all_product_info_by_id(id):
    id = int(id)
    if id < 0:
        return jsonify({'message': 'Fail.'}), 404
    db_product = Product.query.filter(Product.id == id).first()
    db_abstract_product = AbstractProduct.query.filter(AbstractProduct.id == db_product.abstract_product_id).first()
    product_category = ProductCategory.get_by_id(db_abstract_product.product_category_id)
    category_type = CategoryType.get_by_id(product_category.category_type_id)
    results = [db_abstract_product.product_name, product_category.product_category_name,
category_type.category_type_name]
    response = jsonify(results)
    response.status_code = 200
    return response
@app.route(BASE_URL + '/change_product', methods=['POST'])
@csrf.exempt
def change_product():
    db_category_type = CategoryType.get_by_name(request.json['category_type_name'])
    db_product_category = ProductCategory.get_by_name(request.json['product_category_name'])
    if db_product_category is None or db_product_category.category_type_id != db_category_type.id:
        db_product_category = ProductCategory(
            product_category_name=request.json['product_category_name'],
            category_type_id=db_category_type.id

```

```

    )
    manual_session.add(db_product_category)
    manual_session.commit()
db_abstract_product = AbstractProduct.query.filter(
    AbstractProduct.product_name == request.json['abstract_product_name']).first()
if db_abstract_product is None or db_abstract_product.product_category_id != db_product_category.id:
    db_abstract_product = AbstractProduct(
        product_name=request.json['abstract_product_name'],
        product_category_id=db_product_category.id
    )
    manual_session.add(db_abstract_product)
    manual_session.commit()
product = Product.query.filter(Product.id == request.json['id']).first()
product.product_name = request.json['product_name']
product.product_price = request.json['product_price']
product.abstract_product_id = db_abstract_product.id
db.session.commit()
return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/change_abstract_product', methods=['POST'])
@csrf.exempt
def change_abstract_product():
    db_category_type = CategoryType.get_by_name(request.json['category_type_name'])
    db_product_category = ProductCategory.get_by_name(request.json['product_category_name'])
    if db_product_category is None or db_product_category.category_type_id != db_category_type.id:
        db_product_category = ProductCategory(
            product_category_name=request.json['product_category_name'],
            category_type_id=db_category_type.id
        )
        manual_session.add(db_product_category)
        manual_session.commit()
    db_abstract_product = AbstractProduct.query.filter(
        AbstractProduct.product_name == request.json['abstract_product_name']).first()
    if db_abstract_product is None or db_abstract_product.product_category_id != db_product_category.id:
        db_abstract_product = AbstractProduct(
            product_name=request.json['abstract_product_name'],
            product_category_id=db_product_category.id
        )
        manual_session.add(db_abstract_product)
        manual_session.commit()
    product = Product.query.filter(Product.id == request.json['id']).first()
    product.abstract_product_id = db_abstract_product.id
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/change_category_product', methods=['POST'])
@csrf.exempt
def change_category_product():
    db_category_type = CategoryType.get_by_name(request.json['category_type_name'])
    db_product_category = ProductCategory.get_by_name(request.json['product_category_name'])
    if db_product_category is None or db_product_category.category_type_id != db_category_type.id:
        db_product_category = ProductCategory(
            product_category_name=request.json['product_category_name'],
            category_type_id=db_category_type.id
        )
        manual_session.add(db_product_category)
        manual_session.commit()
    db_abstract_product = AbstractProduct.query.filter(
        AbstractProduct.product_name == request.json['abstract_product_name']).first()
    if db_abstract_product is None or db_abstract_product.product_category_id != db_product_category.id:
        db_abstract_product = None
        db_abstract_product = AbstractProduct(
            product_name=request.json['abstract_product_name'],
            product_category_id=db_product_category.id

```

```

    )
    manual_session.add(db_abstract_product)
    manual_session.commit()
db.session.commit()
return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/change_category_type', methods=['POST'])
@csrf.exempt
def change_category_type():
    db_category_type = CategoryType.get_by_name(request.json['category_type_name'])
    db_product_category = ProductCategory.get_by_name(request.json['product_category_name'])
    if db_product_category is None or db_product_category.category_type_id != db_category_type.id:
        db_product_category = ProductCategory(
            product_category_name=request.json['product_category_name'],
            category_type_id=db_category_type.id
        )
        manual_session.add(db_product_category)
        manual_session.commit()
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/change_price', methods=['POST'])
@csrf.exempt
def change_price():
    product = Product.query.filter(Product.id == request.json['id']).first()
    product.product_price = request.json['product_price']
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/remove_product', methods=['POST'])
@csrf.exempt
def remove_product():
    product = Product.query.filter(Product.id == request.json['id']).first()
    db.session.delete(product)
    db.session.commit()
    return jsonify({'message': 'Successfully changed.'}), 200
@app.route(BASE_URL + '/add_product', methods=['POST'])
@csrf.exempt
def add_product():
    product = Product(
        product_name='N',
        product_price=1,
        abstract_product_id=70,
        check_id=request.json['check_id']
    )
    db.session.add(product)
    db.session.commit()
    results = [product.id]
    response = jsonify(results)
    response.status_code = 200
    return response

```

### Модуль project/controllers/home\_controller.py:

```

from flask import send_file
from project.controllers.user_controller import *
@app.route('/')
@app.route('/index')
@app.route('/home')
@app.route('/home/index')
def index():
    return render_template("home/index.html")
@app.route('/about')
@app.route('/home/about')
def about():

```

```

    return render_template("home/about.html", title='О сайте')
@app.route('/privacy_policy')
@app.route('/home/privacy_policy')
@app.route('/privacy-policy')
@app.route('/home/privacy-policy')
def privacy_policy():
    return render_template("home/privacy_policy.html", title='Политика конфиденциальности')
@app.route('/agreement')
@app.route('/home/agreement')
def agreement():
    return render_template("home/agreement.html", title='Пользовательское соглашение')
@app.route("/download_apk", methods=['GET', 'POST'])
@app.route("/download-apk", methods=['GET', 'POST'])
def download_apk():
    return send_file(os.path.normpath("static/apk/app_v.1.0.2.apk"), as_attachment=True)

```

### Модуль project/controllers/support\_controller.py:

```

from project.controllers.user_controller import *
@app.route('/support')
@app.route('/support/')
@app.route('/support/index')
def support_index():
    return render_template("support/index.html")

```

### Модуль project/controllers/user\_controller.py:

```

from __future__ import print_function
import calendar
import random
import uuid
import pytils as pytils
from dateutil.rule import rrule, MONTHLY
from datetime import date, datetime, timedelta
from flask import render_template, flash, request, url_for, send_file, redirect
from flask_login import login_required, current_user
from config import *
from project.forms import EditProfileForm, AddCheckForm, AddManualCheckForm
from project.models.abstract_product import AbstractProduct
from project.models.category_type import CategoryType
from project.models.check import Check
from project.models.organization import Organization
from project.models.product import Product
from project.models.product_category import ProductCategory
from project.models.user import User
from sightengine.client import SightengineClient
from project.util.check_process.check_processing import start_check_process
from project.util.date_process import get_label_week_day_by_date, get_label_week_day_by_double_date
from project.util.export_data import detail_information_excel, detail_information_word
from project.util.image_process.image_processing import start_abbyy
from session import manual_session
BASE_URL = '/user/'
client = SightengineClient({'api_user'}, {'api_secret'})
@app.route(BASE_URL + '<public_id>')
@login_required
def user(public_id):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id == user_.public_id:
        page = request.args.get('page', 1, type=int)
        checks = Check.query. \
            filter(Check.user_id == user_.id). \
            order_by(Check.date_time_of_purchase.desc()). \
            paginate(

```

```

        page, CHECKS_PER_PAGE, False
    )
    current_month = datetime.today().month
    current_year = datetime.today().year
    month_date = str(current_year) + '-' + str(current_month)
    month_day = str(calendar.monthrange(current_year, current_month)[1])
    checks_by_month_date = Check.get_all_by_month_date(user_.id, month_date, month_day)
    spent = 0
    for check in checks_by_month_date:
        products_by_month_date = Product.get_all_check_products(check.id)
        for product in products_by_month_date:
            spent += product.product_price
    if user_.wage is not None:
        left = user_.wage - spent
        wage = user_.wage
    else:
        left = 0
        wage = 0
    spent_today = 0
    for check in Check.get_all_today(user_.id):
        products_by_today = Product.get_all_check_products(check.id)
        for product in products_by_today:
            spent_today += product.product_price
    next_url = url_for('user', public_id=user_.public_id, page=checks.next_num) \
        if checks.has_next else None
    prev_url = url_for('user', public_id=user_.public_id, page=checks.prev_num) \
        if checks.has_prev else None
    return render_template('user/profile.html',
        user=user_,
        wage=wage,
        checks=checks.items,
        next_url=next_url,
        prev_url=prev_url,
        spent=spent,
        left=left,
        spent_today=spent_today
    )
    return render_template('errors/404.html'), 404
@app.route(BASE_URL + '<public_id>' + '<check_id>')
@login_required
def check_details(public_id, check_id):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if not current_user.is_anonymous and current_user.public_id == user_.public_id:
        db_organizations = Organization.get_all()
        organizations = []
        db_abstract_products = AbstractProduct.get_all()
        abstract_products = []
        db_product_categories = ProductCategory.get_all()
        product_categories = []
        db_category_types = CategoryType.get_all()
        category_types = []
        for organization in db_organizations:
            organizations.append(organization.legal_name)
        for abstract_product in db_abstract_products:
            abstract_products.append(abstract_product.product_name)
        for category_type in db_category_types:
            product_categories.append(category_type.category_type_name)
        for product_category in db_product_categories:
            category_types.append(product_category.product_category_name)
        products = Product.query.filter_by(check_id=check_id).all()
        check = Check.query.filter_by(id=check_id).first_or_404()
        organization = Organization.query.filter_by(id=check.organization_id).first_or_404()

```

```

total_cost = 0
for product in products:
    total_cost += product.product_price
form = AddCheckForm()
return render_template('user/check_details.html',
                      check=check,
                      products=products,
                      organization=organization,
                      total_cost=total_cost,
                      abstract_products=abstract_products,
                      product_categories=product_categories,
                      category_types=category_types,
                      user=user_,
                      form=form,
                      )
return render_template('errors/404.html'), 404
@app.route(BASE_URL + 'edit_profile', methods=['GET', 'POST'])
@login_required
def edit_profile():
    form = EditProfileForm()
    if form.validate_on_submit():
        current_user.username = form.username.data
        current_user.wage = form.wage.data
        db.session.commit()
        flash('Your changes have been saved.')
        return user(current_user.public_id)
    elif request.method == 'GET':
        form.username.data = current_user.username
        form.wage.data = current_user.wage
    return render_template('user/edit_profile.html', title='Редактирование профиля', form=form)
@app.route(BASE_URL + '<public_id>/detail_information_all_time')
def detail_information_all_time(public_id):
    return detail_information(public_id, 'all_time')
@app.route(BASE_URL + '<public_id>/detail_information_week')
def detail_information_week(public_id):
    return detail_information(public_id, 'week')
@app.route(BASE_URL + '<public_id>/detail_information_month')
def detail_information_month(public_id):
    return detail_information(public_id, 'month')
@app.route(BASE_URL + '<public_id>/detail_information_year')
def detail_information_year(public_id):
    return detail_information(public_id, 'year')
@app.route(BASE_URL + '<public_id>/<start_date>_<final_date>', methods=['GET', 'POST'])
@csrf.exempt
def detail_information_range(public_id, start_date, final_date):
    return detail_information(public_id, 'range', start_date, final_date)
def detail_information(public_id, sample_type, start_date=None, final_date=None):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    if sample_type == 'all_time':
        legend = 'За все время'
        checks = Check.query.filter(Check.user_id == current_user.id).order_by(Check.date_time_of_purchase.asc()).all()
    elif sample_type == 'week':
        legend = 'За неделю'
        checks = Check.get_all_by_week(user_.id)
    elif sample_type == 'month':
        legend = 'За месяц'
        checks = Check.get_all_by_month(user_.id)
    elif sample_type == 'year':
        legend = 'За год'
        checks = Check.get_all_by_year(user_.id)

```

```

elif sample_type == 'range':
    start_date = datetime.strptime(start_date, '%Y-%m-%d %H:%M:%S')
    final_date = datetime.strptime(final_date, '%Y-%m-%d %H:%M:%S')
    if final_date < start_date:
        start_date, final_date = final_date, start_date
    if start_date.year == final_date.year:
        sample_type = 'month' if start_date.month == final_date.month else 'year'
    legend = '3a ' + str(datetime.date(start_date)) + ' no ' + str(datetime.date(final_date))
    checks = Check.get_all_by_range(user_.id,
                                    start_date,
                                    final_date)

checks_amount = []
dates = []
current_sample_type_date = 0
total_cost = 0
product_amount = 0
dict_category_types = {}
dict_product_categories = {}
dict_abstract_products = {}
max_cost_product = ""
max_cost_price = -1
min_cost_product = ""
min_cost_price = -1
first_month = 1
for check in checks:
    if current_sample_type_date == 0:
        if sample_type == 'all_time':
            current_sample_type_date = check.date_time_of_purchase.year
        elif sample_type == 'week' or sample_type == 'month':
            current_sample_type_date = check.date_time_of_purchase
        elif sample_type == 'year':
            current_sample_type_date = check.date_time_of_purchase.month
            first_month = check.date_time_of_purchase.month
    if sample_type == 'all_time':
        if current_sample_type_date != check.date_time_of_purchase.year:
            checks_amount.append(product_amount)
            dates.append(current_sample_type_date)
            current_sample_type_date = check.date_time_of_purchase.year
            product_amount = 0
    elif sample_type == 'week' or sample_type == 'month':
        if current_sample_type_date != check.date_time_of_purchase:
            checks_amount.append(product_amount)
            dates.append(current_sample_type_date)
            current_sample_type_date = check.date_time_of_purchase
            product_amount = 0
    if sample_type == 'year':
        if current_sample_type_date != check.date_time_of_purchase.month:
            checks_amount.append(product_amount)
            dates.append(current_sample_type_date)
            current_sample_type_date = check.date_time_of_purchase.month
            product_amount = 0
    products = Product.get_all_check_products(check.id)
    for product in products:
        if max_cost_price and min_cost_price == -1:
            max_cost_price = min_cost_price = float(product.product_price)
            min_cost_product = max_cost_product = product.product_name
        if max_cost_price < float(product.product_price):
            max_cost_product = product.product_name
            max_cost_price = product.product_price
        if min_cost_price > float(product.product_price):
            min_cost_product = product.product_name
            min_cost_price = product.product_price

```

```

product_amount += product.product_price
total_cost += product.product_price
abstract_product = AbstractProduct.get_by_id(product.abstract_product_id)
product_category = ProductCategory.get_by_id(abstract_product.product_category_id)
category_type = CategoryType.get_by_id(product_category.category_type_id)
if dict_category_types.get(category_type.category_type_name) is None:
    dict_category_types[category_type.category_type_name] = [{
        product_category.product_category_name: [{
            abstract_product.product_name: product.product_price
        }]}]
else:
    for item in dict_category_types[category_type.category_type_name]:
        for type_product in item.keys():
            if product_category.product_category_name == type_product:
                list_product_category = dict_category_types[category_type.category_type_name].copy()
                for i in list_product_category:
                    if i.get(type_product):
                        list_abstract_product = i[type_product].copy()
                        list_abstract_product.append({
                            abstract_product.product_name: product.product_price
                        })
                        list_product_category.remove(i)
                        for j in i.get(type_product):
                            if j.get(abstract_product.product_name):
                                dict_abstract_product = {
                                    abstract_product.product_name: j[abstract_product.product_name] +
                                    product.product_price
                                }
                                for ij in range(len(list_abstract_product)):
                                    if list_abstract_product[ij].get(abstract_product.product_name):
                                        list_abstract_product[ij] = dict_abstract_product
                                list_abstract_product = [
                                    dict(t) for t in set([tuple(d.items()) for d in list_abstract_product])
                                ]
                                list_product_category_copy = list_product_category.copy()
                                list_product_category_copy.append({type_product: list_abstract_product})
                                dict_category_types[category_type.category_type_name] = list_product_category_copy
            if any(product_category.product_category_name in s.keys()
                for s in dict_category_types[category_type.category_type_name]):
                list_product_category = dict_category_types[category_type.category_type_name].copy()
                list_product_category.append({
                    product_category.product_category_name: [{
                        abstract_product.product_name: product.product_price
                    }]}))
            else:
                list_category_type = dict_category_types[category_type.category_type_name].copy()
                list_category_type.append({
                    product_category.product_category_name: [{
                        abstract_product.product_name: product.product_price
                    }]}))
            if list_category_type is not None:
                dict_category_types[category_type.category_type_name] = list_category_type
if dict_abstract_products.get(abstract_product.product_name) is None:
    dict_abstract_products[abstract_product.product_name] = product.product_price
else:
    dict_abstract_products[abstract_product.product_name] = \
        dict_abstract_products[abstract_product.product_name] + product.product_price
values_pie = []
labels_pie = []
category_type_cost = 0
values_radar = []

```



```

type_product_cost = 0
last_category_type = ""
last_product_category = ""
total_costs_category_type = {}
total_costs_product_category = {}
for category_type in dict_category_types.keys():
    labels_pie.append(category_type)
    if category_type_cost != 0:
        total_costs_category_type[last_category_type] = category_type_cost
        values_pie.append(category_type_cost)
        category_type_cost = 0
    print(category_type)
    last_category_type = category_type
    for product_categories in dict_category_types.values():
        if dict_category_types[category_type] == product_categories:
            for list_product_category in product_categories:
                for product_category in list_product_category:
                    if type_product_cost != 0:
                        total_costs_product_category[last_product_category] = type_product_cost
                        values_radar.append(tuple((last_category_type, last_product_category, type_product_cost)))
                        type_product_cost = 0
                    print(product_category)
                    last_product_category = product_category
                    for abstract_products in list_product_category.values():
                        if abstract_products == list_product_category[product_category]:
                            for dict_abstract_product in abstract_products:
                                for abstract_product in dict_abstract_product:
                                    print(abstract_product)
                                    print(dict_abstract_product[abstract_product])
                                    category_type_cost += float(dict_abstract_product[abstract_product])
                                    type_product_cost += float(dict_abstract_product[abstract_product])
if current_sample_type_date != 0:
    checks_amount.append(product_amount)
    dates.append(current_sample_type_date)
if sample_type == 'week':
    counter = 0
    new_dates = []
    new_amounts = []
    for i in range(7):
        new_dates.append(date.today() + timedelta(days=-date.today().weekday() + i))
    if len(dates) != 7:
        for i in range(7):
            if (len(dates) == counter or datetime.date(dates[counter]) != date.today() +
                timedelta(days=-date.today().weekday() + i)) and date.today() >= \
                date.today() + timedelta(days=-date.today().weekday() + i):
                new_amounts.append(0)
            elif date.today() < date.today() + timedelta(days=-date.today().weekday() + i):
                pass
            else:
                new_amounts.append(checks_amount[counter])
                counter += 1
    checks_amount = new_amounts
    dates = new_dates
    for i in range(len(dates)):
        dates[i] = get_label_week_day_by_date(dates[i])
elif sample_type == 'month':
    counter = 0
    new_dates = []
    new_amounts = []
    if len(dates) == 0:
        if start_date is not None:
            dates.append(start_date)

```

```

else:
    dates.append(datetime.today())
    checks_amount.append(0)
days_in_month = calendar.monthrange(datetime.today().year, datetime.today().month)[1]
if start_date is not None:
    days_in_month = final_date.day - start_date.day + 1
for i in range(days_in_month):
    if start_date is not None:
        new_dates.append(i + start_date.day)
    else:
        new_dates.append(i + 1)
if len(dates) != days_in_month:
    for i in range(days_in_month):
        if start_date is not None and (len(dates) == counter or dates[counter].day != i + start_date.day) \
            and date.today().day > i + start_date.day:
            new_amounts.append(0)
        elif start_date is not None and date.today().day < i + start_date.day:
            pass
        elif start_date is not None:
            new_amounts.append(checks_amount[counter])
            counter += 1
        elif (len(dates) == counter or dates[counter].day != i + 1) and date.today().day > i:
            new_amounts.append(0)
        elif date.today().day <= i:
            pass
        else:
            new_amounts.append(checks_amount[counter])
            counter += 1
    checks_amount = new_amounts
    dates = new_dates
elif sample_type == 'year':
    counter = 0
    new_amounts = []
    month_in_year = 12
    if start_date is not None:
        month_in_year = final_date.month - start_date.month + 1
    if len(dates) != month_in_year:
        for i in range(month_in_year):
            if start_date is not None and (len(dates) == counter or dates[counter] != i + start_date.month) \
                and date.today().month > i + start_date.month:
                new_amounts.append(0)
            elif start_date is not None and date.today().month < i + start_date.month:
                pass
            elif start_date is not None and len(dates) != counter:
                new_amounts.append(checks_amount[counter])
                counter += 1
            elif (len(dates) == counter or dates[counter] != i + 1) and date.today().month > i:
                new_amounts.append(0)
            elif date.today().month <= i:
                pass
            else:
                new_amounts.append(checks_amount[counter])
                counter += 1
        checks_amount = new_amounts
        dates = []
    for i in range(month_in_year):
        if start_date is not None:
            dates.append(pytils.dt.ru_strftime(u"%b", inflected=True,
                                                date=date(datetime.today().year, i + start_date.month, 1))
                        + ' (' + str(i + start_date.month) + ')')
        else:
            dates.append(pytils.dt.ru_strftime(u"%b", inflected=True,

```

```

        date=date(datetime.today().year, i + 1, 1))
    + ' (' + str(i + 1) + ')')
values_pie.append(category_type_cost)
values_radar.append(tuple((last_category_type, last_product_category, type_product_cost)))
total_costs_category_type[last_category_type] = category_type_cost
total_costs_product_category[last_product_category] = type_product_cost
result_wage = 0
if user_.wage is None:
    result_wage = 'Укажите Вашу заработную плату'
    result_month = 1
elif len(dates) > 0 and sample_type == 'all_time':
    result_month = len([dt for dt in rrule(MONTHLY,
        dtstart=datetime(dates[0], first_month, 1),
        until=datetime(datetime.now().year, datetime.now().month, 1))]) - 2

    print(result_month)
    result_wage = user_.wage * result_month
else:
    result_wage = user_.wage
    result_month = 1
max_cost_category_type = -1
max_cost_category_type_name = ""
min_cost_category_type = -1
min_cost_category_type_name = ""
for item in total_costs_category_type:
    if max_cost_category_type == -1:
        max_cost_category_type = min_cost_category_type = total_costs_category_type[item]
        min_cost_category_type_name = max_cost_category_type_name = item
    if max_cost_category_type < total_costs_category_type[item]:
        max_cost_category_type_name = item
        max_cost_category_type = total_costs_category_type[item]
    if min_cost_category_type > total_costs_category_type[item]:
        min_cost_category_type_name = item
        min_cost_category_type = total_costs_category_type[item]
print(total_costs_category_type)
max_cost_product_category = -1
max_cost_product_category_name = ""
min_cost_product_category = -1
min_cost_product_category_name = ""
for item in total_costs_product_category:
    if max_cost_product_category == -1:
        max_cost_product_category = min_cost_product_category = total_costs_product_category[item]
        min_cost_product_category_name = max_cost_product_category_name = item
    if max_cost_product_category < total_costs_product_category[item]:
        max_cost_product_category_name = item
        max_cost_product_category = total_costs_product_category[item]
    if min_cost_category_type > total_costs_product_category[item]:
        min_cost_product_category_name = item
        min_cost_product_category = total_costs_product_category[item]
interested_facts = [max_cost_product, max_cost_price, min_cost_product, min_cost_price, result_wage,
result_month,
    max_cost_category_type_name, max_cost_category_type, min_cost_category_type_name,
    min_cost_category_type, max_cost_product_category_name, max_cost_product_category,
    min_cost_product_category_name, min_cost_product_category, ]
return render_template('user/detail_information.html',
    user=user_,
    total_cost=total_cost,
    values=checks_amount,
    labels=dates,
    legend=legend,
    dict_abstract_products=dict_abstract_products,
    dict_category_types=dict_category_types,
    interested_facts=interested_facts,

```

```

        labels_pie=labels_pie,
        values_pie=values_pie,
        values_radar=values_radar,
        total_costs_category_type=total_costs_category_type,
        total_costs_product_category=total_costs_product_category,
        sample_type=sample_type,
    )
@app.route(BASE_URL + '<public_id>' + 'add-check', methods=['GET', 'POST'])
@csrf.exempt
def add_check(public_id):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    form = AddCheckForm()
    db_organizations = Organization.get_all()
    organizations = []
    db_abstract_products = AbstractProduct.get_all()
    abstract_products = []
    db_product_categories = ProductCategory.get_all()
    product_categories = []
    db_category_types = CategoryType.get_all()
    category_types = []
    for organization in db_organizations:
        organizations.append(organization.legal_name)
    for abstract_product in db_abstract_products:
        abstract_products.append(abstract_product.product_name)
    for category_type in db_category_types:
        product_categories.append(category_type.category_type_name)
    for product_category in db_product_categories:
        category_types.append(product_category.product_category_name)
    if request.method == 'POST':
        if 'date_time' in request.form:
            add_manual_check(current_user.public_id)
            return render_template('user/add_check.html',
                                   form=form,
                                   dateTime=datetime.now(),
                                   organizations=organizations,
                                   abstract_products=abstract_products,
                                   product_categories=product_categories,
                                   category_types=category_types)
    files = request.files.getlist('file[]')
    filename = request.files['file[]']
    if filename.filename == "":
        return render_template('user/add_check.html',
                               form=form,
                               dateTime=datetime.now(),
                               organizations=organizations,
                               abstract_products=abstract_products,
                               product_categories=product_categories,
                               category_types=category_types)
    for file in files:
        dynamic_name = str(uuid.uuid4())
        file_name = DIRECTORY_TEMP_FILES + dynamic_name
        source_file = file_name + '.jpg'
        target_file = file_name + '.xml'
        file.save(source_file)
        start_abbyy(target_file, source_file)
        start_check_process(target_file, current_user.public_id)
        os.remove(source_file)
        os.remove(target_file)
    return render_template('user/add_check.html',
                           form=form,

```

```

        dateTime=datetime.now(),
        organizations=organizations,
        abstract_products=abstract_products,
        product_categories=product_categories,
        category_types=category_types)
elif request.method == 'GET':
    return render_template('user/add_check.html',
        form=form,
        dateTime=datetime.now(),
        organizations=organizations,
        abstract_products=abstract_products,
        product_categories=product_categories,
        category_types=category_types)
@app.route(BASE_URL + '<public_id>' + 'add-check', methods=['GET', 'POST'])
@csrf.exempt
def add_manual_check(public_id):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    date_time = request.form.get('date_time')
    organization = request.form.get('organization')
    abstract_products = request.form.getlist('abstract_product[]')
    category_types = request.form.getlist('category_type[]')
    product_categories = request.form.getlist('product_category[]')
    product_names = request.form.getlist('product_name[]')
    product_price = request.form.getlist('product_price[]')
    if organization is None or organization == "":
        db_organization = Organization.query.filter(Organization.legal_name == 'Неопределено').first()
    else:
        db_organization = Organization.query.filter(Organization.legal_name == organization).first()
    if db_organization is None:
        db_organization = Organization(legal_name=organization)
        manual_session.add(db_organization)
        manual_session.commit()
    db_check = Check(
        date_time_of_purchase=date_time,
        organization_id=db_organization.id,
        user_id=current_user.id
    )
    manual_session.add(db_check)
    manual_session.commit()
    product_amount = len(abstract_products)
    for i in range(product_amount):
        db_category_type = CategoryType.get_by_name(product_categories[i])
        db_product_category = ProductCategory.get_by_name(category_types[i])
        if db_product_category is None or db_product_category.category_type_id != db_category_type.id:
            db_product_category = ProductCategory(
                product_category_name=category_types[i],
                category_type_id=db_category_type.id
            )
            manual_session.add(db_product_category)
            manual_session.commit()
        db_abstract_product = AbstractProduct.query.filter(AbstractProduct.product_name == abstract_products[i]).first()
        if db_abstract_product is None or db_abstract_product.product_category_id != db_product_category.id:
            db_abstract_product = AbstractProduct(
                product_name=abstract_products[i],
                product_category_id=db_product_category.id
            )
            manual_session.add(db_abstract_product)
            manual_session.commit()
        db_product = Product(
            product_name=product_names[i],

```

```

        product_price=product_price[i],
        check_id=db_check.id,
        abstract_product_id=db_abstract_product.id
    )
    manual_session.add(db_product)
    manual_session.commit()
@app.route(BASE_URL + '<public_id>/<check_id>/excel', methods=['GET', 'POST'])
@csrf.exempt
def check_excel(public_id, check_id):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    date_time = request.form.get('date_time')
    organization = request.form.get('organization')
    abstract_products = request.form.getlist('abstract_product[]')
    category_types = request.form.getlist('category_type[]')
    product_categories = request.form.getlist('product_category[]')
    product_names = request.form.getlist('product_name[]')
    product_price = request.form.getlist('product_price[]')
    if request.form.get('word') == 'on':
        temp = []
        total_price = 0
        for i in range(len(abstract_products)):
            buf = {}
            buf['abstract_product'] = abstract_products[i]
            buf['category_type'] = category_types[i]
            buf['product_category'] = product_categories[i]
            buf['product_name'] = product_names[i]
            buf['product_price'] = product_price[i]
            total_price += float(product_price[i])
            temp.append(buf)
        from mailmerge import MailMerge
        word_name = check_id + ".docx"
        template_1 = "template.docx"
        if organization is None:
            organization = ''
        cust = {
            'organization': organization,
            'date': date_time,
            'total_cost': str('{:.2f}'.format(total_price)),
        }
        document_3 = MailMerge(template_1)
        document_3.merge(**cust)
        document_3.merge_rows('product_name', temp)
        document_3.write(word_name)
        return send_file(word_name, as_attachment=True)
    excel_name = check_id + ".xls"
    dict_product_categories = {}
    dict_category_type = {}
    total_price = 0
    for num in range(0, len(abstract_products)):
        total_price += float(product_price[num])
        if dict_product_categories.get(product_categories[num]) is None:
            dict_product_categories[product_categories[num]] = float(product_price[num])
        else:
            dict_product_categories[product_categories[num]] = \
                dict_product_categories[product_categories[num]] + float(product_price[num])
        if dict_category_type.get(category_types[num]) is None:
            dict_category_type[category_types[num]] = float(product_price[num])
        else:
            dict_category_type[category_types[num]] = \
                dict_category_type[category_types[num]] + float(product_price[num])

```

```

result_price = [float(item) for item in product_price]
import pandas as pd
df1 = pd.DataFrame({'№': [c + 1 for c in range(len(abstract_products))])
df2 = pd.DataFrame({'Название продукта': product_names})
df3 = pd.DataFrame({'Цена': result_price})
df4 = pd.DataFrame({'Продукт': abstract_products})
df5 = pd.DataFrame({'Тип': category_types})
df6 = pd.DataFrame({'Категория': product_categories})
df7 = pd.DataFrame({'Дата': [date_time]})
df8 = pd.DataFrame({'Организация': [organization]})
df9 = pd.DataFrame({'Итого': [total_price]})
df10 = pd.DataFrame({'Тип': [c for c in dict_category_type.keys()]})
df11 = pd.DataFrame({'Цена': [c for c in dict_category_type.values()]})
df12 = pd.DataFrame({'Категория': [c for c in dict_product_categories.keys()]})
df13 = pd.DataFrame({'Цена': [c for c in dict_product_categories.values()]})
writer = pd.ExcelWriter(excel_name, engine='xlsxwriter')
df1.to_excel(writer, sheet_name='Sheet1', startrow=19, index=False)
df2.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=1, index=False)
df3.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=2, index=False)
df4.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=3, index=False)
df5.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=4, index=False)
df6.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=5, index=False)
df7.to_excel(writer, sheet_name='Sheet1', startcol=1, index=False)
df8.to_excel(writer, sheet_name='Sheet1', startcol=3, index=False)
df9.to_excel(writer, sheet_name='Sheet1', startcol=2, index=False)
df10.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=4, index=False)
df11.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=5, index=False)
df12.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=1, index=False)
df13.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=2, index=False)
workbook = writer.book
worksheet = writer.sheets['Sheet1']
chart1 = workbook.add_chart({'type': 'pie',
                             'subtype': 'smooth'})
chart1.add_series({'categories': ['Sheet1', 20, 3, len(df2) + 19, 3],
                  'values': ['Sheet1', 20, 2, len(df2) + 19, 2]})
worksheet.insert_chart('A4', chart1)
chart2 = workbook.add_chart({'type': 'pie',
                             'subtype': 'smooth'})
chart2.add_series({'categories': [
    'Sheet1',
    22 + len(abstract_products),
    4,
    len(df2) + 21 + len(dict_category_type),
    4,
],
                  'values': [
    'Sheet1',
    22 + len(abstract_products),
    5,
    len(df2) + 21 + len(dict_category_type),
    5,
]})
worksheet.insert_chart('D4', chart2)
chart3 = workbook.add_chart({'type': 'pie',
                             'subtype': 'smooth'})
chart3.add_series({'categories': [
    'Sheet1',
    22 + len(abstract_products),
    1,
    len(df2) + 21 + len(dict_product_categories),
    1,
],
                  'values': [
    'Sheet1',
    22 + len(abstract_products),
    1,
    len(df2) + 21 + len(dict_product_categories),
    1,
]})

```

```

'values': [
    'Sheet1',
    22 + len(abstract_products),
    2,
    len(df2) + 21 + len(dict_product_categories),
    2,
    ]})
worksheet.insert_chart('F4', chart3)
cell_format = workbook.add_format({'italic': True})
column_num_format = workbook.add_format()
column_num_format.set_num_format('#,##0.00')
table_format = workbook.add_format()
table_format.set_border(6)
table_format.set_border_color('white')
table_format.set_font_color('gray')
header_format = workbook.add_format()
after_header_format = workbook.add_format()
header_format.set_pattern(1) # This is optional when using a solid fill.
header_format.set_bg_color('black')
header_format.set_align('center')
header_format.set_align('vcenter')
header_format.set_font_color('white')
header_format.set_border(6)
header_format.set_border_color('white')
header_format.set_font_name('Times New Roman')
header_format.set_font_size(14)
after_header_format.set_align('center')
after_header_format.set_align('vcenter')
after_header_format.set_bg_color('gray')
after_header_format.set_border(13)
after_header_format.set_border_color('black')
after_header_format.set_font_name('Times New Roman')
after_header_format.set_font_size(12)
worksheet.write('B1', 'Дара', header_format)
worksheet.write('C1', 'Итого', header_format)
worksheet.write('D1', 'Организация', header_format)
worksheet.write('B2', date_time, after_header_format)
worksheet.write('C2', total_price, after_header_format)
worksheet.write('D2', organization, after_header_format)
worksheet.set_row(0, 21, table_format)
worksheet.set_column('A:A', 3, cell_format)
worksheet.set_column('B:B', 52)
worksheet.set_column('C:C', 10, column_num_format)
worksheet.set_column('D:E', 33)
worksheet.set_column('F:F', 40)
writer.save()
return send_file(excel_name, as_attachment=True)
@app.route(BASE_URL + '<public_id>/excel_all_time', methods=['GET', 'POST'])
@csrf.exempt
def detail_excel_all_time(public_id):
    return detail_information_excel(public_id, 'all_time')
@app.route(BASE_URL + '<public_id>/excel_week', methods=['GET', 'POST'])
@csrf.exempt
def detail_excel_week(public_id):
    return detail_information_excel(public_id, 'week')
@app.route(BASE_URL + '<public_id>/excel_month', methods=['GET', 'POST'])
@csrf.exempt
def detail_excel_month(public_id):
    return detail_information_excel(public_id, 'month')
@app.route(BASE_URL + '<public_id>/excel_year', methods=['GET', 'POST'])
@csrf.exempt
def detail_excel_year(public_id):

```



```

    return detail_information_excel(public_id, 'year')
@app.route(BASE_URL + '/<public_id>/excel_range', methods=['GET', 'POST'])
@csrf.exempt
def detail_range(public_id):
    start_date = request.form.get('start_date')
    final_date = request.form.get('final_date')
    if request.form.get('sample_type') == 'on':
        return redirect(url_for('detail_information_range',
                                public_id=public_id,
                                sample_type='range',
                                start_date=datetime.strptime(start_date, '%Y-%m-%d'),
                                final_date=datetime.strptime(final_date, '%Y-%m-%d'),
                                ))
    if request.form.get('word') == 'on':
        return detail_information_word(public_id, [start_date, final_date])
    return detail_information_excel(public_id, [start_date, final_date])
@app.route(BASE_URL + '<public_id>/word_all_time', methods=['GET', 'POST'])
@csrf.exempt
def detail_word_all_time(public_id):
    return detail_information_word(public_id, 'all_time')
@app.route(BASE_URL + '<public_id>/word_week', methods=['GET', 'POST'])
@csrf.exempt
def detail_word_week(public_id):
    return detail_information_word(public_id, 'week')
@app.route(BASE_URL + '<public_id>/word_month', methods=['GET', 'POST'])
@csrf.exempt
def detail_word_month(public_id):
    return detail_information_word(public_id, 'month')
@app.route(BASE_URL + '<public_id>/word_year', methods=['GET', 'POST'])
@csrf.exempt
def detail_word_year(public_id):
    return detail_information_word(public_id, 'year')
@app.route(BASE_URL + '<public_id>/compare_information_week', methods=['GET', 'POST'])
@csrf.exempt
def compare_information_week(public_id):
    return compare_information(public_id, 'week')
@app.route(BASE_URL + '<public_id>/compare_information_month', methods=['GET', 'POST'])
@csrf.exempt
def compare_information_month(public_id):
    return compare_information(public_id, 'month')
@app.route(BASE_URL + '<public_id>/compare_information_year', methods=['GET', 'POST'])
@csrf.exempt
def compare_information_year(public_id):
    return compare_information(public_id, 'year')
@app.route(BASE_URL + '<public_id>/compare_range', methods=['GET', 'POST'])
@csrf.exempt
def compare_information_range(public_id):
    start_date_first_period = request.form.get('start_date_first_period')
    final_date_first_period = request.form.get('final_date_first_period')
    start_date_second_period = request.form.get('start_date_second_period')
    final_date_second_period = request.form.get('final_date_second_period')
    return compare_information(
        public_id=public_id,
        sample_type='range',
        start_date_first_period=start_date_first_period,
        final_date_first_period=final_date_first_period,
        start_date_second_period=start_date_second_period,
        final_date_second_period=final_date_second_period,
    )
def compare_information(public_id, sample_type, start_date_first_period=None, final_date_first_period=None,
                        start_date_second_period=None, final_date_second_period=None):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()

```

```

if current_user.is_anonymous or current_user.public_id != user_.public_id:
    return render_template('errors/404.html'), 404
if sample_type == 'week':
    legend_first_period = 'За текущую неделю'
    legend_second_period = 'За прошлую неделю'
    checks = Check.get_all_by_double_week(user_.id)
    start_date_second_period = date.today() - timedelta(days=date.today().weekday() + 7)
    final_date_second_period = date.today() - timedelta(days=date.today().weekday())
    start_date_first_period = date.today() - timedelta(days=date.today().weekday())
    final_date_first_period = date.today() + timedelta(days=-date.today().weekday() + 7)
elif sample_type == 'month':
    legend_first_period = 'За текущий месяц'
    legend_second_period = 'За прошлый месяц'
    checks = Check.get_all_by_double_month(user_.id)
    if datetime.now().month == 1:
        last_period_month = 12
        last_period_year = datetime.today().year - 1
    else:
        last_period_month = datetime.today().month
        last_period_year = datetime.today().year
    start_date_second_period = date.today() - timedelta(
        days=date.today().day + calendar.monthrange(last_period_year, last_period_month)[1])
    final_date_second_period = date.today() - timedelta(days=date.today().day)
    start_date_first_period = date.today() - timedelta(days=date.today().day - 1)
    final_date_first_period = date.today() + timedelta(
        days=calendar.monthrange(last_period_year, last_period_month)[1] - date.today().day)
elif sample_type == 'year':
    legend_first_period = 'За текущий год'
    legend_second_period = 'За прошлый год'
    checks = Check.get_all_by_double_year(user_.id)
    start_date_second_period = datetime.date(datetime.strptime(str(datetime.now().year - 1) + '-01-01', '%Y-%m-%d'))
    final_date_second_period = datetime.date(datetime.strptime(str(datetime.now().year - 1) + '-12-31', '%Y-%m-%d'))
    start_date_first_period = datetime.date(datetime.strptime(str(datetime.now().year) + '-01-01', '%Y-%m-%d'))
    final_date_first_period = datetime.date(datetime.strptime(str(datetime.now().year) + '-12-31', '%Y-%m-%d'))
elif sample_type == 'range':
    start_date_first_period = datetime.date(datetime.strptime(start_date_first_period, '%Y-%m-%d'))
    final_date_first_period = datetime.date(datetime.strptime(final_date_first_period, '%Y-%m-%d'))
    start_date_second_period = datetime.date(datetime.strptime(start_date_second_period, '%Y-%m-%d'))
    final_date_second_period = datetime.date(datetime.strptime(final_date_second_period, '%Y-%m-%d'))
    if final_date_first_period < start_date_first_period:
        start_date_first_period, final_date_first_period = final_date_first_period, start_date_first_period
    if start_date_first_period.year == final_date_first_period.year and \
        start_date_second_period.year == final_date_second_period.year:
        sample_type = 'month' if start_date_first_period.month == final_date_first_period.month and \
            start_date_second_period.month == final_date_second_period.month else 'year'
    legend_first_period = 'За ' + str(start_date_first_period) + \
        ' по ' + str(final_date_first_period)
    legend_second_period = 'За ' + str(start_date_second_period) + \
        ' по ' + str(final_date_second_period)
    checks = Check.get_all_by_double_range(user_.id,
        start_date_first_period,
        final_date_first_period,
        start_date_second_period,
        final_date_second_period, )
print('start_date_f_p ', start_date_first_period, 'final_date_f_p ', final_date_first_period)
print('start_date_s_p ', start_date_second_period, 'final_date_s_p ', final_date_second_period)
checks_amount = []
dates = []
current_sample_type_date = 0
total_cost = 0
product_amount = 0

```

```

dict_category_types = {}
dict_product_categories = {}
dict_abstract_products = {}
max_cost_product = ""
max_cost_price = -1
min_cost_product = ""
min_cost_price = -1
first_month = 1
for check in checks:
    if current_sample_type_date == 0:
        current_sample_type_date = check.date_time_of_purchase
        first_month = check.date_time_of_purchase.month
    if sample_type == 'week' or sample_type == 'month':
        if current_sample_type_date != check.date_time_of_purchase:
            checks_amount.append(product_amount)
            dates.append(current_sample_type_date)
            current_sample_type_date = check.date_time_of_purchase
            product_amount = 0
    if sample_type == 'year':
        if current_sample_type_date.month != check.date_time_of_purchase.month or \
            current_sample_type_date.year != check.date_time_of_purchase.year:
            checks_amount.append(product_amount)
            dates.append(current_sample_type_date)
            current_sample_type_date = check.date_time_of_purchase
            product_amount = 0
    products = Product.get_all_check_products(check.id)
    for product in products:
        if max_cost_price and min_cost_price == -1:
            max_cost_price = min_cost_price = float(product.product_price)
            min_cost_product = max_cost_product = product.product_name
        if max_cost_price < float(product.product_price):
            max_cost_product = product.product_name
            max_cost_price = product.product_price
        if min_cost_price > float(product.product_price):
            min_cost_product = product.product_name
            min_cost_price = product.product_price
        product_amount += product.product_price
        total_cost += product.product_price
        abstract_product = AbstractProduct.get_by_id(product.abstract_product_id)
        product_category = ProductCategory.get_by_id(abstract_product.product_category_id)
        category_type = CategoryType.get_by_id(product_category.category_type_id)
        if dict_category_types.get(category_type.category_type_name) is None:
            if datetime.date(check.date_time_of_purchase) >= start_date_first_period:
                dict_category_types[category_type.category_type_name] = [{
                    product_category.product_category_name: [{
                        abstract_product.product_name: product.product_price
                    }]
                }]
            else:
                dict_category_types[category_type.category_type_name] = [{
                    product_category.product_category_name: [{
                        '2' + abstract_product.product_name: product.product_price
                    }]
                }]
        else:
            for item in dict_category_types[category_type.category_type_name]:
                for type_product in item.keys():
                    if product_category.product_category_name == type_product:
                        list_product_category = dict_category_types[category_type.category_type_name].copy()
                        for i in list_product_category:
                            if i.get(type_product):
                                list_abstract_product = i[type_product].copy()

```

```

        if datetime.date(check.date_time_of_purchase) >= start_date_first_period:
            list_abstract_product.append({
                abstract_product.product_name: product.product_price
            })
        else:
            list_abstract_product.append({
                '2' + abstract_product.product_name: product.product_price
            })
        list_product_category.remove(i)
        for j in i.get(type_product):
            if j.get(abstract_product.product_name):
                dict_abstract_product = {
                    abstract_product.product_name: j[abstract_product.product_name] +
                    product.product_price
                }
                for ij in range(len(list_abstract_product)):
                    if list_abstract_product[ij].get(abstract_product.product_name):
                        list_abstract_product[ij] = dict_abstract_product
                list_abstract_product = [
                    dict(t) for t in set([tuple(d.items()) for d in list_abstract_product])
                ]
            if j.get('2' + abstract_product.product_name):
                dict_abstract_product = {
                    '2' + abstract_product.product_name:
                    j['2' + abstract_product.product_name] + product.product_price
                }
                for ij in range(len(list_abstract_product)):
                    if list_abstract_product[ij].get('2' + abstract_product.product_name):
                        list_abstract_product[ij] = dict_abstract_product
                list_abstract_product = [
                    dict(t) for t in set([tuple(d.items()) for d in list_abstract_product])
                ]
            list_product_category_copy = list_product_category.copy()
            list_product_category_copy.append({type_product: list_abstract_product})
            dict_category_types[category_type.category_type_name] = list_product_category_copy
    if any(product_category.product_category_name in s.keys()
        for s in dict_category_types[category_type.category_type_name]):
        list_product_category = dict_category_types[category_type.category_type_name].copy()
    if datetime.date(check.date_time_of_purchase) >= start_date_first_period:
        list_product_category.append({
            product_category.product_category_name: [{
                abstract_product.product_name: product.product_price
            }]
        })
    else:
        list_product_category.append({
            product_category.product_category_name: [{
                '2' + abstract_product.product_name: product.product_price
            }]
        })
    else:
        list_category_type = dict_category_types[category_type.category_type_name].copy()

    if datetime.date(check.date_time_of_purchase) >= start_date_first_period:
        list_category_type.append({
            product_category.product_category_name: [{
                abstract_product.product_name: product.product_price
            }]
        })
    else:
        list_category_type.append({
            product_category.product_category_name: [{
                '2' + abstract_product.product_name: product.product_price
            }]
        })
    if list_category_type is not None:

```

```

        dict_category_types[category_type.category_type_name] = list_category_type
    if dict_abstract_products.get(abstract_product.product_name) is None:
        dict_abstract_products[abstract_product.product_name] = product.product_price
    else:
        dict_abstract_products[abstract_product.product_name] = \
            dict_abstract_products[abstract_product.product_name] + product.product_price
    if dict_abstract_products.get('2' + abstract_product.product_name) is None:
        dict_abstract_products['2' + abstract_product.product_name] = product.product_price
    else:
        dict_abstract_products['2' + abstract_product.product_name] = \
            dict_abstract_products['2' + abstract_product.product_name] + product.product_price
total_cost_first_period = 0
total_cost_second_period = 0
values_pie = []
values_pie_first_period = []
values_pie_second_period = []
labels_pie = []
category_type_cost = 0
category_type_cost_first_period = 0
category_type_cost_second_period = 0
values_radar = []
values_radar_first_period = []
values_radar_second_period = []
type_product_cost = 0
type_product_cost_first_period = 0
type_product_cost_second_period = 0
last_category_type = ""
last_category_type_first_period = ""
last_category_type_second_period = ""
last_product_category = ""
last_type_product_first_period = ""
last_type_product_second_period = ""
total_costs_category_type = {}
total_costs_product_category = {}
max_cost_category_type_first_period = -1
max_cost_category_type_name_first_period = ""
min_cost_category_type_first_period = -1
min_cost_category_type_name_first_period = ""
max_cost_category_type_second_period = -1
max_cost_category_type_name_second_period = ""
min_cost_category_type_second_period = -1
min_cost_category_type_name_second_period = ""
max_cost_type_product_first_period = -1
max_cost_type_product_name_first_period = ""
min_cost_type_product_first_period = -1
min_cost_type_product_name_first_period = ""
max_cost_type_product_second_period = -1
max_cost_type_product_name_second_period = ""
min_cost_type_product_second_period = -1
min_cost_type_product_name_second_period = ""
for category_type in dict_category_types.keys():
    labels_pie.append(category_type)
    if category_type_cost != 0:
        total_costs_category_type[last_category_type] = category_type_cost
        values_pie.append(category_type_cost)
        category_type_cost = 0
    if category_type_cost_first_period != 0:
        values_pie_first_period.append(category_type_cost_first_period)
    if max_cost_category_type_first_period == -1:
        max_cost_category_type_first_period = min_cost_category_type_first_period = \
            category_type_cost_first_period
        max_cost_category_type_name_first_period = min_cost_category_type_name_first_period = \

```

```

        last_category_type_first_period
    if max_cost_category_type_first_period < category_type_cost_first_period:
        max_cost_category_type_first_period = category_type_cost_first_period
        max_cost_category_type_name_first_period = last_category_type_first_period
    if min_cost_category_type_first_period > category_type_cost_first_period:
        min_cost_category_type_first_period = category_type_cost_first_period
        min_cost_category_type_name_first_period = last_category_type_first_period
    category_type_cost_first_period = 0
if category_type_cost_second_period != 0:
    values_pie_second_period.append(category_type_cost_second_period)
if max_cost_category_type_second_period == -1:
    max_cost_category_type_second_period = min_cost_category_type_second_period = \
        category_type_cost_second_period
    max_cost_category_type_name_second_period = min_cost_category_type_name_second_period = \
        last_category_type_second_period
if max_cost_category_type_second_period < category_type_cost_second_period:
    max_cost_category_type_second_period = category_type_cost_second_period
    max_cost_category_type_name_second_period = last_category_type_second_period
if min_cost_category_type_second_period > category_type_cost_second_period:
    min_cost_category_type_second_period = category_type_cost_second_period
    min_cost_category_type_name_second_period = last_category_type_second_period
category_type_cost_second_period = 0
print(category_type)
last_category_type = category_type
for product_categories in dict_category_types.values():
    if dict_category_types[category_type] == product_categories:
        for list_product_category in product_categories:
            for product_category in list_product_category:
                if type_product_cost != 0:
                    total_costs_product_category[last_product_category] = type_product_cost
                    values_radar.append(tuple((last_category_type, last_product_category, type_product_cost)))
                    type_product_cost = 0
                if type_product_cost_first_period != 0:
                    values_radar_first_period.append(tuple((last_category_type_first_period,
                                                            last_type_product_first_period,
                                                            type_product_cost_first_period)))
                    type_product_cost = 0
                if max_cost_type_product_first_period == -1:
                    max_cost_type_product_first_period = min_cost_type_product_first_period = \
                        type_product_cost_first_period
                    max_cost_type_product_name_first_period = min_cost_type_product_name_first_period = \
                        last_type_product_first_period
                if max_cost_type_product_first_period < type_product_cost_first_period:
                    max_cost_type_product_first_period = type_product_cost_first_period
                    max_cost_type_product_name_first_period = last_type_product_first_period
                if min_cost_type_product_first_period > type_product_cost_first_period:
                    min_cost_type_product_first_period = type_product_cost_first_period
                    min_cost_type_product_name_first_period = last_type_product_first_period
                type_product_cost_first_period = 0
            elif type_product_cost_first_period == 0:
                values_radar_first_period.append(tuple((last_category_type,
                                                         last_product_category,
                                                         0)))
        if type_product_cost_second_period != 0:
            values_radar_second_period.append(tuple((last_category_type_second_period,
                                                      last_type_product_second_period,
                                                      type_product_cost_second_period)))
        if max_cost_type_product_second_period == -1:
            max_cost_type_product_second_period = min_cost_type_product_second_period = \
                type_product_cost_second_period
            max_cost_type_product_name_second_period = min_cost_type_product_name_second_period = \
                last_type_product_second_period

```

```

if max_cost_type_product_second_period < type_product_cost_second_period:
    max_cost_type_product_second_period = type_product_cost_second_period
    max_cost_type_product_name_second_period = last_type_product_second_period
if min_cost_type_product_second_period > type_product_cost_second_period:
    min_cost_type_product_second_period = type_product_cost_second_period
    min_cost_type_product_name_second_period = last_type_product_second_period
type_product_cost_second_period = 0
elif type_product_cost_second_period == 0:
    values_radar_second_period.append(tuple((last_category_type,
                                             last_product_category,
                                             0)))

print(product_category)
last_product_category = product_category
for abstract_products in list_product_category.values():
    if abstract_products == list_product_category[product_category]:
        for dict_abstract_product in abstract_products:
            for abstract_product in dict_abstract_product:
                print(abstract_product)
                print(dict_abstract_product[abstract_product])
                category_type_cost += float(dict_abstract_product[abstract_product])
                type_product_cost += float(dict_abstract_product[abstract_product])
                if abstract_product[0] == '2':
                    total_cost_second_period += float(dict_abstract_product[abstract_product])
                    last_category_type_second_period = category_type
                    category_type_cost_second_period += \
                        float(dict_abstract_product[abstract_product])
                    last_type_product_second_period = product_category
                    type_product_cost_second_period += \
                        float(dict_abstract_product[abstract_product])
                else:
                    total_cost_first_period += float(dict_abstract_product[abstract_product])
                    last_category_type_first_period = category_type
                    category_type_cost_first_period += \
                        float(dict_abstract_product[abstract_product])
                    last_type_product_first_period = product_category
                    type_product_cost_first_period += \
                        float(dict_abstract_product[abstract_product])
if type_product_cost_first_period != 0:
    if max_cost_type_product_first_period == -1:
        max_cost_type_product_first_period = min_cost_type_product_first_period = \
            type_product_cost_first_period
        max_cost_type_product_name_first_period = min_cost_type_product_name_first_period = \
            last_type_product_first_period
    if max_cost_type_product_first_period < type_product_cost_first_period:
        max_cost_type_product_first_period = type_product_cost_first_period
        max_cost_type_product_name_first_period = last_type_product_first_period
    if min_cost_type_product_first_period > type_product_cost_first_period:
        min_cost_type_product_first_period = type_product_cost_first_period
        min_cost_type_product_name_first_period = last_type_product_first_period
if type_product_cost_second_period != 0:
    if max_cost_type_product_second_period == -1:
        max_cost_type_product_second_period = min_cost_type_product_second_period = \
            type_product_cost_second_period
        max_cost_type_product_name_second_period = min_cost_type_product_name_second_period = \
            last_type_product_second_period
    if max_cost_type_product_second_period < type_product_cost_second_period:
        max_cost_type_product_second_period = type_product_cost_second_period
        max_cost_type_product_name_second_period = last_type_product_second_period
    if min_cost_type_product_second_period > type_product_cost_second_period:
        min_cost_type_product_second_period = type_product_cost_second_period
        min_cost_type_product_name_second_period = last_type_product_second_period
if category_type_cost_first_period != 0:

```

```

if max_cost_category_type_first_period == -1:
    max_cost_category_type_first_period = min_cost_category_type_first_period = \
        category_type_cost_first_period
    max_cost_category_type_name_first_period = min_cost_category_type_name_first_period = \
        last_category_type_first_period
if max_cost_category_type_first_period < category_type_cost_first_period:
    max_cost_category_type_first_period = category_type_cost_first_period
    max_cost_category_type_name_first_period = last_category_type_first_period
if min_cost_category_type_first_period > category_type_cost_first_period:
    min_cost_category_type_first_period = category_type_cost_first_period
    min_cost_category_type_name_first_period = last_category_type_first_period
if category_type_cost_second_period != 0:
    if max_cost_category_type_second_period == -1:
        max_cost_category_type_second_period = min_cost_category_type_second_period = \
            category_type_cost_second_period
        max_cost_category_type_name_second_period = min_cost_category_type_name_second_period = \
            last_category_type_second_period
    if max_cost_category_type_second_period < category_type_cost_second_period:
        max_cost_category_type_second_period = category_type_cost_second_period
        max_cost_category_type_name_second_period = last_category_type_second_period
    if min_cost_category_type_second_period > category_type_cost_second_period:
        min_cost_category_type_second_period = category_type_cost_second_period
        min_cost_category_type_name_second_period = last_category_type_second_period
if current_sample_type_date != 0:
    checks_amount.append(product_amount)
    dates.append(current_sample_type_date)
if sample_type == 'week':
    counter = 0
    new_dates = []
    checks_amount_first_period = []
    checks_amount_second_period = []
    for i in range(7):
        new_dates.append(date.today() + timedelta(days=-date.today().weekday() + i))
    if len(dates) != 14:
        for i in range(7):
            if datetime.date(dates[counter]) < date.today() - timedelta(days=date.today().weekday() + 1):
                counter += 1
            else:
                break
        for i in range(7):
            print(checks_amount[counter])
            if (datetime.date(dates[counter]) != date.today() +
                timedelta(days=-date.today().weekday() + i)) and date.today() >= \
                date.today() + timedelta(days=-date.today().weekday() + i):
                checks_amount_first_period.append(0)
            elif date.today() < date.today() + timedelta(days=-date.today().weekday() + i):
                pass
            else:
                checks_amount_first_period.append(checks_amount[counter])
                counter += 1
        preview_counter = counter
        counter = 0
        for i in range(7):
            if preview_counter == counter or datetime.date(dates[counter]) != date.today() + \
                timedelta(days=-date.today().weekday() + i - 7) and date.today() >= \
                date.today() + timedelta(days=-date.today().weekday() + i):
                checks_amount_second_period.append(0)
            elif date.today() < date.today() + timedelta(days=-date.today().weekday() + i - 7):
                pass
            else:
                checks_amount_second_period.append(checks_amount[counter])
                counter += 1

```



```

    dates = new_dates
    for i in range(len(dates)):
        dates[i] = get_label_week_day_by_double_date(dates[i])
elif sample_type == 'month':
    counter = 0
    new_dates = []
    checks_amount_first_period = []
    checks_amount_second_period = []
    if datetime.now().month == 1:
        last_period_month = 12
        last_period_year = datetime.today().year - 1
    else:
        last_period_month = datetime.today().month - 1
        last_period_year = datetime.today().year
    days_in_month_first_period = calendar.monthrange(datetime.today().year, datetime.today().month)[1]
    days_in_month_second_period = calendar.monthrange(last_period_year, last_period_month)[1]
    print(start_date_first_period)
    if days_in_month_first_period > days_in_month_second_period:
        for i in range(days_in_month_first_period):
            new_dates.append(i + 1)
    else:
        for i in range(days_in_month_second_period):
            new_dates.append(i + 1)
    if len(dates) != days_in_month_first_period + days_in_month_second_period:
        for i in range(len(dates)):
            if datetime.date(dates[counter]).month < date.today().month:
                counter += 1
            else:
                break
        for i in range(days_in_month_first_period):
            if len(dates) == counter or datetime.date(dates[counter]).day != i + 1 and date.today().day > i:
                checks_amount_first_period.append(0)
            elif date.today().day < i + 1:
                pass
            else:
                checks_amount_first_period.append(checks_amount[counter])
                counter += 1
        preview_counter = counter
        counter = 0
        for i in range(days_in_month_second_period):
            if preview_counter == counter or datetime.date(dates[counter]).day != i + 1:
                checks_amount_second_period.append(0)
            else:
                checks_amount_second_period.append(checks_amount[counter])
                counter += 1
    dates = new_dates
elif sample_type == 'year':
    counter = 0
    checks_amount_first_period = []
    checks_amount_second_period = []
    month_in_year = 12
    if len(dates) != month_in_year * 2:
        for i in range(len(dates)):
            if datetime.date(dates[counter]).year != date.today().year:
                counter += 1
            else:
                break
        for i in range(month_in_year):
            if len(dates) == counter or datetime.date(dates[counter]).month != i + 1 and date.today().month > i:
                checks_amount_first_period.append(0)
            elif date.today().month < i + 1:
                pass

```

```

else:
    checks_amount_first_period.append(checks_amount[counter])
    counter += 1
preview_counter = counter
counter = 0
for i in range(month_in_year):
    if preview_counter == counter or datetime.date(dates[counter]).month != i + 1:
        checks_amount_second_period.append(0)
    else:
        checks_amount_second_period.append(checks_amount[counter])
        counter += 1
dates = []
for i in range(month_in_year):
    dates.append(pytils.dt.ru_strftime(u"%b", inflected=True,
                                      date=date(datetime.today().year, i + 1, 1))
                + ' (' + str(i + 1) + ')')
values_pie.append(category_type_cost)
values_pie_first_period.append(category_type_cost_first_period)
values_pie_second_period.append(category_type_cost_second_period)
print(values_pie_first_period)
print(values_pie_second_period)
values_radar.append(tuple((last_category_type, last_product_category, type_product_cost)))
values_radar_first_period.append(tuple((last_category_type_first_period,
                                       last_type_product_first_period,
                                       type_product_cost_first_period)))
values_radar_second_period.append(tuple((last_category_type_second_period,
                                       last_type_product_second_period,
                                       type_product_cost_second_period)))
total_costs_category_type[last_category_type] = category_type_cost
total_costs_product_category[last_product_category] = type_product_cost
result_wage = 0
if user_.wage is None:
    result_wage = 'Укажите Вашу заработную плату'
    result_month = 1
elif len(dates) > 0 and sample_type == 'all_time':
    result_month = len([dt for dt in rrule(MONTHLY,
                                          dtstart=datetime(dates[0], first_month, 1),
                                          until=datetime(datetime.now().year, datetime.now().month, 1))]) - 2
    print(result_month)
    result_wage = user_.wage * result_month
else:
    result_wage = user_.wage
    result_month = 1
max_cost_category_type = -1
max_cost_category_type_name = ""
min_cost_category_type = -1
min_cost_category_type_name = ""
for item in total_costs_category_type:
    if max_cost_category_type == -1:
        max_cost_category_type = min_cost_category_type = total_costs_category_type[item]
        min_cost_category_type_name = max_cost_category_type_name = item
    if max_cost_category_type < total_costs_category_type[item]:
        max_cost_category_type_name = item
        max_cost_category_type = total_costs_category_type[item]
    if min_cost_category_type > total_costs_category_type[item]:
        min_cost_category_type_name = item
        min_cost_category_type = total_costs_category_type[item]
print(total_costs_category_type)
max_cost_product_category = -1
max_cost_product_category_name = ""
min_cost_product_category = -1
min_cost_product_category_name = ""

```

```

for item in total_costs_product_category:
    if max_cost_product_category == -1:
        max_cost_product_category = min_cost_product_category = total_costs_product_category[item]
        min_cost_product_category_name = max_cost_product_category_name = item
    if max_cost_product_category < total_costs_product_category[item]:
        max_cost_product_category_name = item
        max_cost_product_category = total_costs_product_category[item]
    if min_cost_category_type > total_costs_product_category[item]:
        min_cost_product_category_name = item
        min_cost_product_category = total_costs_product_category[item]
    interested_facts = [max_cost_product, max_cost_price, min_cost_product, min_cost_price, result_wage,
result_month,
                        max_cost_category_type_name, max_cost_category_type, min_cost_category_type_name,
                        min_cost_category_type, max_cost_product_category_name, max_cost_product_category,
                        min_cost_product_category_name, min_cost_product_category, total_cost_first_period,
                        total_cost_second_period, max_cost_category_type_first_period,
                        max_cost_category_type_name_first_period, min_cost_category_type_first_period,
                        min_cost_category_type_name_first_period, max_cost_category_type_second_period,
                        max_cost_category_type_name_second_period, min_cost_category_type_second_period,
                        min_cost_category_type_name_second_period, max_cost_type_product_first_period,
                        max_cost_type_product_name_first_period, min_cost_type_product_first_period,
                        min_cost_type_product_name_first_period, max_cost_type_product_second_period,
                        max_cost_type_product_name_second_period, min_cost_type_product_second_period,
                        min_cost_type_product_name_second_period]
return render_template('user/compare_information.html',
                        user=user_,
                        total_cost=total_cost,
                        values=checks_amount,
                        checks_amount_first_period=checks_amount_first_period,
                        checks_amount_second_period=checks_amount_second_period,
                        labels=dates,
                        legend_first_period=legend_first_period,
                        legend_second_period=legend_second_period,
                        dict_abstract_products=dict_abstract_products,
                        dict_category_types=dict_category_types,
                        interested_facts=interested_facts,
                        labels_pie=labels_pie,

                        values_pie=values_pie,
                        values_pie_first_period=values_pie_first_period,
                        values_pie_second_period=values_pie_second_period,

                        values_radar=values_radar,
                        values_radar_first_period=values_radar_first_period,
                        values_radar_second_period=values_radar_second_period,

                        total_costs_category_type=total_costs_category_type,
                        total_costs_product_category=total_costs_product_category,
                        sample_type=sample_type,
                        )

```

## Модуль project/util/check\_process/check\_processing.py:

```

from datetime import datetime
from project.models.abstract_product import AbstractProduct
from project.models.check import Check
from project.models.organization import Organization
from project.models.product import Product
from project.models.user import User
from project.util.check_process.check_parser import DataCheck, check_parser, product_parser,
highlight_abstract_product
from session import manual_session

```

```

# Direct check processing
def start_check_process(target_file, user_public_id):
    global data_check
    data_check = DataCheck()
    file_process(target_file, data_check)
    data_base_process(user_public_id)
def file_process(target_file, data_check):
    previous_line = ""
    text = ""
    f = open(target_file, 'rt')
    data_check.Products = {}
    data_check.AbstractProducts = []
    for line in f:
        line = ' '.join(line.split())
        line = check_parser(line, data_check)
        upper_text = line.upper()
        if upper_text.find('МАШИПЫТ') != -1 or upper_text.find('ЭКСПЕСС') != -1:
            data_check.Products['Машипыт'] = 0
        if data_check.Products.get('Машипыт'):
            if data_check.ResultPrice != 0:
                data_check.Products['Машипыт'] = data_check.ResultPrice
        else:
            line = product_parser(line, previous_line, data_check)
            text += line
            previous_line = line
    f.close()
def data_base_process(user_public_id):
    if data_check.Date == "":
        data_check.Date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    try:
        if int(str(data_check.Date)[0:4]) > datetime.now().year:
            data_check.Date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    except ValueError as err:
        print(err)
    db_organization = Organization.get_organization(data_check.LegalName, data_check.LegalAddress)
    if db_organization is None:
        if data_check.LegalName is None or data_check.LegalName == "":
            db_organization = Organization.query.filter(Organization.legal_name == 'Неопределено').first()
        else:
            if data_check.Unp == "":
                data_check.Unp = 11111111
            if len(data_check.LegalAddress) > 49:
                data_check.LegalAddress = data_check.LegalAddress[0:49]
            db_organization = Organization(
                legal_name=data_check.LegalName,
                legal_address=data_check.LegalAddress,
                taxpayer_identification_number=data_check.Unp)
            manual_session.add(db_organization)
            manual_session.commit()
    db_user = User.get_user_by_public_id(user_public_id)
    db_check = Check(
        date_time_of_purchase=data_check.Date,
        organization_id=db_organization.id,
        user_id=db_user.id
    )
    manual_session.add(db_check)
    manual_session.commit()
    if data_check.Products.get('Машипыт'):
        db_abstract_product = AbstractProduct.get_product('Машипыт')
        db_product = Product(
            product_name='Машипыт',
            product_price=data_check.ResultPrice,

```

```

        check_id=db_check.id,
        abstract_product_id=db_abstract_product.id
    )
    manual_session.add(db_product)
    manual_session.commit()
else:
    for product_name in data_check.Products:
        if len(product_name) < 2:
            continue
        abstract_product = highlight_abstract_product(product_name)
        abstract_product = abstract_product.capitalize()
        db_abstract_product = AbstractProduct.get_product(abstract_product)
        if db_abstract_product is None:
            temp_str = abstract_product[0]
            i = 1
            while len(abstract_product) > i and abstract_product[i].islower():
                abstract_product = abstract_product[i]
                i += 1
            db_abstract_product = AbstractProduct.get_product(temp_str)
        if db_abstract_product is None:
            part_len = int(len(abstract_product) / 2)
            db_abstract_product = AbstractProduct.get_product(abstract_product[:part_len])
            if db_abstract_product is None:
                db_abstract_product = AbstractProduct.get_product('Неопределено')
        current_price = float(data_check.Products[product_name])
        db_product = Product(
            product_name=product_name,
            product_price=current_price,
            check_id=db_check.id,
            abstract_product_id=db_abstract_product.id
        )
        manual_session.add(db_product)
        manual_session.commit()

```

### Модуль project/util/export\_data.py:

```

import datetime
from flask import send_file, redirect, url_for, render_template
from flask_login import current_user
from project.models.abstract_product import AbstractProduct
from project.models.category_type import CategoryType
from project.models.check import Check
from project.models.product import Product
from project.models.product_category import ProductCategory
from project.models.user import User
def detail_information_word(public_id, type_select):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    if len(type_select) == 2:
        if datetime.datetime.strptime(type_select[0], '%Y-%m-%d') > \
            datetime.datetime.strptime(type_select[1], '%Y-%m-%d'):
            type_select[0], type_select[1] = type_select[1], type_select[0]
        checks = Check.get_all_by_range(
            user_.id, datetime.datetime.strptime(type_select[0], '%Y-%m-%d'),
            datetime.datetime.strptime(type_select[1], '%Y-%m-%d')
        )
        word_name = type_select[0] + "_" + type_select[1] + ".docx"
        title_label = "3a " + type_select[0] + " по " + type_select[1]
    elif type_select == 'all_time':
        checks = Check.query.filter(Check.user_id == current_user.id).order_by(Check.date_time_of_purchase.asc()).all()
        word_name = "all_time.docx"

```

```

        title_label = 'За все время'
    elif type_select == 'week':
        checks = Check.get_all_by_week(user_.id)
        word_name = "week.docx"
        title_label = 'За неделю'
    elif type_select == "month":
        month_day = str(datetime.datetime.now().day + 1)
        current_date = str(datetime.datetime.now().year) + '-' + str(datetime.datetime.now().month)
        checks = Check.get_all_by_month_date(user_.id, current_date, month_day)
        word_name = "month.docx"
        title_label = 'За месяц'
    elif type_select == 'year':
        checks = Check.get_all_by_year(user_.id)
        word_name = "year.docx"
        title_label = 'За год'
    else:
        return render_template('errors/404.html'), 404
    total_price = 0
    dict_product_categories = {}
    dict_category_type = {}
    product_price = []
    abstract_products = []
    product_names = []
    category_types = []
    product_categories = []
    for check in checks:
        products = Product.get_all_check_products(check.id)
        for product in products:
            abstract_product = AbstractProduct.get_by_id(product.abstract_product_id)
            product_category = ProductCategory.get_by_id(abstract_product.product_category_id)
            category_type = CategoryType.get_by_id(product_category.category_type_id)
            total_price += float(product.product_price)
            product_price.append(product.product_price)
            abstract_products.append(abstract_product.product_name)
            product_names.append(product.product_name)
            category_types.append(product_category.product_category_name)
            product_categories.append(category_type.category_type_name)
            if dict_product_categories.get(category_type.category_type_name) is None:
                dict_product_categories[category_type.category_type_name] = float(product.product_price)
            else:
                dict_product_categories[category_type.category_type_name] = \
                    dict_product_categories[category_type.category_type_name] + float(product.product_price)
            if dict_category_type.get(product_category.product_category_name) is None:
                dict_category_type[product_category.product_category_name] = float(product.product_price)
            else:
                dict_category_type[product_category.product_category_name] = \
                    dict_category_type[product_category.product_category_name] + float(product.product_price)
    result_price = [float(item) for item in product_price]
    if len(abstract_products) < 1:
        return render_template('errors/403.html'), 403
    temp = []
    total_price = 0
    for i in range(len(abstract_products)):
        buf = {}
        buf['abstract_product'] = abstract_products[i]
        buf['category_type'] = category_types[i]
        buf['product_category'] = product_categories[i]
        buf['product_name'] = product_names[i]
        buf['product_price'] = str(product_price[i])
        total_price += float(product_price[i])
        temp.append(buf)
    from mailmerge import MailMerge

```

```

template_1 = "template_detail_information.docx"
cust = {
    'total_cost': str('{:.2f}'.format(total_price)),
}
document_3 = MailMerge(template_1)
document_3.merge(**cust)
document_3.merge_rows('product_name', temp)
document_3.write(word_name)
return send_file(word_name, as_attachment=True)
def detail_information_excel(public_id, type_select):
    user_ = User.query.filter_by(public_id=public_id).first_or_404()
    if current_user.is_anonymous or current_user.public_id != user_.public_id:
        return render_template('errors/404.html'), 404
    if len(type_select) == 2:
        if datetime.datetime.strptime(type_select[0], '%Y-%m-%d') > \
            datetime.datetime.strptime(type_select[1], '%Y-%m-%d'):
            type_select[0], type_select[1] = type_select[1], type_select[0]
        checks = Check.get_all_by_range(
            user_.id, datetime.datetime.strptime(type_select[0], '%Y-%m-%d'),
            datetime.datetime.strptime(type_select[1], '%Y-%m-%d')
        )
        excel_name = type_select[0] + "_" + type_select[1] + ".xls"
        title_label = "За " + type_select[0] + " по " + type_select[1]
    elif type_select == 'all_time':
        checks = Check.query.filter(Check.user_id == current_user.id).order_by(Check.date_time_of_purchase.asc()).all()
        excel_name = "all_time.xls"
        title_label = 'За все время'
    elif type_select == 'week':
        checks = Check.get_all_by_week(user_.id)
        excel_name = "week.xls"
        title_label = 'За неделю'
    elif type_select == "month":
        month_day = str(datetime.datetime.now().day + 1)
        current_date = str(datetime.datetime.now().year) + '-' + str(datetime.datetime.now().month)
        checks = Check.get_all_by_month_date(user_.id, current_date, month_day)
        excel_name = "month.xls"
        title_label = 'За месяц'
    elif type_select == 'year':
        checks = Check.get_all_by_year(user_.id)
        excel_name = "year.xls"
        title_label = 'За год'
    else:
        return render_template('errors/404.html'), 404
total_price = 0
dict_product_categories = {}
dict_category_type = {}
product_price = []
abstract_products = []
product_names = []
category_types = []
product_categories = []
for check in checks:
    products = Product.get_all_check_products(check.id)
    for product in products:
        abstract_product = AbstractProduct.get_by_id(product.abstract_product_id)
        product_category = ProductCategory.get_by_id(abstract_product.product_category_id)
        category_type = CategoryType.get_by_id(product_category.category_type_id)
        total_price += float(product.product_price)
        product_price.append(product.product_price)
        abstract_products.append(abstract_product.product_name)
        product_names.append(product.product_name)
        category_types.append(product_category.product_category_name)

```

```

product_categories.append(category_type.category_type_name)
if dict_product_categories.get(category_type.category_type_name) is None:
    dict_product_categories[category_type.category_type_name] = float(product.product_price)
else:
    dict_product_categories[category_type.category_type_name] = \
        dict_product_categories[category_type.category_type_name] + float(product.product_price)
if dict_category_type.get(product_category.product_category_name) is None:
    dict_category_type[product_category.product_category_name] = float(product.product_price)
else:
    dict_category_type[product_category.product_category_name] = \
        dict_category_type[product_category.product_category_name] + float(product.product_price)
result_price = [float(item) for item in product_price]
if len(abstract_products) < 1:
    return render_template('errors/403.html'), 403
import pandas as pd
df1 = pd.DataFrame({'№': [c + 1 for c in range(len(abstract_products))])
df2 = pd.DataFrame({'Название продукта': product_names})
df3 = pd.DataFrame({'Цена': result_price})
df4 = pd.DataFrame({'Продукт': abstract_products})
df5 = pd.DataFrame({'Тип': category_types})
df6 = pd.DataFrame({'Категория': product_categories})
df7 = pd.DataFrame({'За все время'})
df9 = pd.DataFrame({'Итого': [total_price]})
df10 = pd.DataFrame({'Тип': [c for c in dict_category_type.keys()]})
df11 = pd.DataFrame({'Цена': [c for c in dict_category_type.values()]})
df12 = pd.DataFrame({'Категория': [c for c in dict_product_categories.keys()]})
df13 = pd.DataFrame({'Цена': [c for c in dict_product_categories.values()]})
writer = pd.ExcelWriter(excel_name, engine='xlsxwriter')
df1.to_excel(writer, sheet_name='Sheet1', startrow=19, index=False)
df2.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=1, index=False)
df3.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=2, index=False)
df4.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=3, index=False)
df5.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=4, index=False)
df6.to_excel(writer, sheet_name='Sheet1', startrow=19, startcol=5, index=False)
df7.to_excel(writer, sheet_name='Sheet1', startcol=1, index=False)
df9.to_excel(writer, sheet_name='Sheet1', startcol=2, index=False)
df10.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=4, index=False)
df11.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=5, index=False)
df12.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=1, index=False)
df13.to_excel(writer, sheet_name='Sheet1', startrow=21 + len(abstract_products), startcol=2, index=False)
workbook = writer.book
worksheet = writer.sheets['Sheet1']
chart1 = workbook.add_chart({'type': 'pie',
                             'subtype': 'smooth'})
chart1.add_series({'categories': ['Sheet1', 20, 3, len(df2) + 19, 3],
                  'values': ['Sheet1', 20, 2, len(df2) + 19, 2]})
worksheet.insert_chart('A4', chart1)
chart2 = workbook.add_chart({'type': 'pie',
                             'subtype': 'smooth'})
chart2.add_series({'categories': [
    'Sheet1',
    22 + len(abstract_products),
    4,
    len(df2) + 21 + len(dict_category_type),
    4,
],
                  'values': [
    'Sheet1',
    22 + len(abstract_products),
    5,
    len(df2) + 21 + len(dict_category_type),
    5,

```



```

    })
    worksheet.insert_chart('D4', chart2)

    chart3 = workbook.add_chart({'type': 'pie',
                                'subtype': 'smooth'})
    chart3.add_series({'categories': [
        'Sheet1',
        22 + len(abstract_products),
        1,
        len(df2) + 21 + len(dict_product_categories),
        1,
    ],
        'values': [
            'Sheet1',
            22 + len(abstract_products),
            2,
            len(df2) + 21 + len(dict_product_categories),
            2,
        ]})
    worksheet.insert_chart('F4', chart3)
    cell_format = workbook.add_format({'italic': True})
    column_num_format = workbook.add_format()
    column_num_format.set_num_format('#,##0.00')
    table_format = workbook.add_format()
    table_format.set_border(6)
    table_format.set_border_color('white')
    table_format.set_font_color('gray')
    header_format = workbook.add_format()
    after_header_format = workbook.add_format()
    header_format.set_pattern(1) # This is optional when using a solid fill.
    header_format.set_bg_color('black')
    header_format.set_align('center')
    header_format.set_align('vcenter')
    header_format.set_font_color('white')
    header_format.set_border(6)
    header_format.set_border_color('white')
    header_format.set_font_name('Times New Roman')
    header_format.set_font_size(14)
    after_header_format.set_align('center')
    after_header_format.set_align('vcenter')
    after_header_format.set_bg_color('gray')
    after_header_format.set_border(13)
    after_header_format.set_border_color('black')
    after_header_format.set_font_name('Times New Roman')
    after_header_format.set_font_size(12)
    worksheet.write('B1', 'Дата', header_format)
    worksheet.write('C1', 'Итого', header_format)
    worksheet.write('D1', 'Организация', header_format)
    worksheet.write('B2', title_label, after_header_format)
    worksheet.write('C2', total_price, after_header_format)
    worksheet.set_row(0, 21, table_format)
    worksheet.set_column('A:A', 3, cell_format)
    worksheet.set_column('B:B', 52)
    worksheet.set_column('C:C', 10, column_num_format)
    worksheet.set_column('D:E', 33)
    worksheet.set_column('F:F', 40)
    writer.save()
    return send_file(excel_name, as_attachment=True)

```

Модуль project/models/user.py:

```
from hashlib import md5
```

```

from time import time
import jwt
from flask_login import UserMixin
from werkzeug.security import generate_password_hash, check_password_hash
from config import *
class User(UserMixin, db.Model):
    """ User Model for storing user related details """
    __tablename__ = "user"
    id = db.Column(db.INTEGER, primary_key=True)
    email = db.Column(db.String(255), unique=True, nullable=False)
    registered_on = db.Column(db.DateTime, nullable=False)
    is_admin = db.Column(db.Boolean, nullable=False, default=False)
    public_id = db.Column(db.String(100), unique=True)
    username = db.Column(db.String(50), unique=True)
    password_hash = db.Column(db.String(100))
    confirmed = db.Column(db.Boolean, nullable=False, default=False)
    confirmed_on = db.Column(db.DateTime, nullable=True)
    wage = db.Column(db.NUMERIC(15, 2), unique=False, nullable=True)
    @property
    def password(self):
        raise AttributeError('password: write-only field')
    @password.setter
    def password(self, password):
        self.password_hash = generate_password_hash(password)
    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
    def get_id(self):
        return str(self.id)
    @staticmethod
    def get_user_by_public_id(public_id):
        return User.query.filter(User.public_id == public_id).first()
    def avatar(self, size):
        digest = md5(self.email.lower().encode('utf-8')).hexdigest()
        return 'https://www.gravatar.com/avatar/{ }?d=identicon&s={ }'.format(digest, size)
    def link_avatar(self):
        return md5(self.email.lower().encode('utf-8'))
    def get_reset_password_token(self, expires_in=600):
        return jwt.encode(
            {'reset_password': self.id, 'exp': time() + expires_in},
            app.config['SECRET_KEY'], algorithm='HS256').decode('utf-8')
    @staticmethod
    def verify_reset_password_token(token):
        try:
            id_ = jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])['reset_password']
        except:
            return
        return User.query.get(id_)
    def get_confirm_email_token(self, expires_in=3600):
        return jwt.encode(
            {'confirm_email': self.id, 'exp': time() + expires_in},
            app.config['SECRET_KEY'], algorithm='HS256').decode('utf-8')
    @staticmethod
    def verify_confirm_email_token(token):
        try:
            id_ = jwt.decode(token, app.config['SECRET_KEY'], algorithms=['HS256'])['confirm_email']
        except:
            return
        return User.query.get(id_)
    def __repr__(self):
        return "<User '{ }>".format(self.username)
    @login.user_loader
    def load_user(id_):

```

```
return User.query.get(int(id_))
```

### Модуль project/models/product\_category.py:

```
from config import *
class ProductCategory(db.Model):
    __tablename__ = 'product_category'
    id = db.Column(db.Integer, primary_key=True)
    product_category_name = db.Column(db.String(80), unique=True, nullable=False)
    category_type_id = db.Column(db.Integer, db.ForeignKey('category_type.id'))
    def __repr__(self):
        return '<ProductCategory %r>' % self.product_category_name
    @staticmethod
    def get_all():
        return ProductCategory.query.all()
    @staticmethod
    def get_by_id(id):
        return ProductCategory.query.filter(ProductCategory.id == id).first()
    @staticmethod
    def get_by_name(name):
        return ProductCategory.query.filter(ProductCategory.product_category_name.contains(name)).first()
```

### Модуль project/models/product.py:

```
from config import *
class Product(db.Model):
    __tablename__ = 'product'
    id = db.Column(db.Integer, primary_key=True)
    product_name = db.Column(db.String(50), unique=False, nullable=True)
    product_price = db.Column(db.NUMERIC(15, 2), unique=False, nullable=True)
    check_id = db.Column(db.Integer, db.ForeignKey('check.id'))
    abstract_product_id = db.Column(db.Integer, db.ForeignKey('abstract_product.id'))
    def __repr__(self):
        return '<Product %r>' % self
    @staticmethod
    def get_all():
        return Product.query.all()
    @staticmethod
    def get_all_check_products(check_id):
        return Product.query.filter(Product.check_id == check_id).all()
    @staticmethod
    def get_product_by_id(product_id):
        return Product.query.filter(Product.id == product_id).first()
```

### Модуль project/models/organization.py:

```
from config import *
class Organization(db.Model):
    __tablename__ = 'organization'
    id = db.Column(db.Integer, primary_key=True)
    legal_name = db.Column(db.String(50), unique=False, nullable=True)
    legal_address = db.Column(db.String(50), unique=False, nullable=True)
    taxpayer_identification_number = db.Column(db.INTEGER, unique=False, nullable=True)
    def __repr__(self):
        return '<Organization %r>' % self.legal_name
    @staticmethod
    def get_all():
        return Organization.query.all()
    @staticmethod
    def get_organization(name, address):
        return Organization.query.filter(Organization.legal_name.contains(name),
                                         Organization.legal_address.contains(address)).first()
```

## Модуль project/models/check.py:

```
from datetime import datetime, timedelta
from config import *
from sqlalchemy import or_, and_
class Check(db.Model):
    __tablename__ = 'check'
    id = db.Column(db.Integer, primary_key=True)
    date_time_of_purchase = db.Column(db.DateTime, unique=False, nullable=True)
    organization_id = db.Column(db.Integer, db.ForeignKey('organization.id'))
    user_id = db.Column(db.INTEGER, db.ForeignKey('user.id'))
    def __repr__(self):
        return '<Check %r>' % self.date_time_of_purchase
    @staticmethod
    def get_all():
        return Check.query.all()
    @staticmethod
    def get_all_user_checks(user_id):
        return Check.query.filter(Check.user_id == user_id).all()
    @staticmethod
    def get_by_id(user_id):
        return Check.query.filter(Check.id == user_id).first()
    @staticmethod
    def get_all_by_week(user_id):
        weekday = datetime.today().weekday()
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(
                                       datetime.date(datetime.now()) - timedelta(days=weekday), datetime.now())
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_by_month_date(user_id, date, month_day):
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(date + '-01', date + '-' + month_day)
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_by_month(user_id):
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(
                                       str(datetime.now().year) + '-' +
                                       str(datetime.now().month) + '-01', datetime.now())
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_by_year(user_id):
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(
                                       str(datetime.now().year) + '-01-01', datetime.now())
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_today(user_id):
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(datetime.date(datetime.now()), datetime.now())
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_by_range(user_id, start_date, final_date):
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(start_date, final_date)
                                   ).order_by(Check.date_time_of_purchase.asc()).all()
    @staticmethod
    def get_all_by_double_week(user_id):
        weekday = datetime.today().weekday()
        return Check.query.filter(Check.user_id == user_id,
                                   Check.date_time_of_purchase.between(
```

```

        datetime.date(datetime.now()) - timedelta(days=weekday + 7), datetime.now())
    ).order_by(Check.date_time_of_purchase.asc()).all()

@staticmethod
def get_all_by_double_month(user_id):
    if datetime.now().month == 1:
        datetime_last_period = str(datetime.now().year - 1) + '-12-01'
    else:
        datetime_last_period = str(datetime.now().year) + '-' + str(datetime.now().month - 1) + '-01'
    return Check.query.filter(Check.user_id == user_id,
                             Check.date_time_of_purchase.between(
                                 datetime_last_period, datetime.now())
                             ).order_by(Check.date_time_of_purchase.asc()).all()

@staticmethod
def get_all_by_double_year(user_id):
    return Check.query.filter(Check.user_id == user_id,
                             Check.date_time_of_purchase.between(
                                 str(datetime.now().year - 1) + '-01-01', datetime.now())
                             ).order_by(Check.date_time_of_purchase.asc()).all()

@staticmethod
def get_all_by_double_range(user_id,
                             start_date_first_period,
                             final_date_first_period,
                             start_date_second_period,
                             final_date_second_period,
                             ):
    return Check.query.filter(or_(and_(Check.user_id == user_id,
                                       Check.date_time_of_purchase.between(start_date_first_period,
                                                                           final_date_first_period)),
                                and_(Check.user_id == user_id,
                                       Check.date_time_of_purchase.between(start_date_second_period,
                                                                           final_date_second_period))))
    ).order_by(Check.date_time_of_purchase.asc()).all()

```

### Модуль project/models/category\_type.py:

```

from config import *
class CategoryType(db.Model):
    __tablename__ = 'category_type'
    id = db.Column(db.Integer, primary_key=True)
    category_type_name = db.Column(db.String(80), unique=True, nullable=False)
    def __repr__(self):
        return '<CategoryType %r>' % self.category_type_name
    @staticmethod
    def get_all():
        return CategoryType.query.all()
    @staticmethod
    def get_by_id(id):
        return CategoryType.query.filter(CategoryType.id == id).first()
    @staticmethod
    def get_by_name(name):
        return CategoryType.query.filter(CategoryType.category_type_name.contains(name)).first()

```

### Модуль project/models/abstract\_product.py:

```

from config import *
class AbstractProduct(db.Model):
    __tablename__ = 'abstract_product'
    id = db.Column(db.Integer, primary_key=True)
    product_name = db.Column(db.String(80), nullable=False)
    product_category_id = db.Column(db.Integer, db.ForeignKey('product_category.id'))
    def __repr__(self):
        return '<AbstractProduct %r>' % self.product_name
    @staticmethod

```

```
def get_all():
    return AbstractProduct.query.all()
    @staticmethod
def get_product(name):
    return AbstractProduct.query.filter(AbstractProduct.product_name.contains(name)).first()
    @staticmethod
def get_by_id(abstract_product_id):
    return AbstractProduct.query.filter(AbstractProduct.id == abstract_product_id).first()
```

## ПРИЛОЖЕНИЕ Б

(справочное)

### Экранные формы

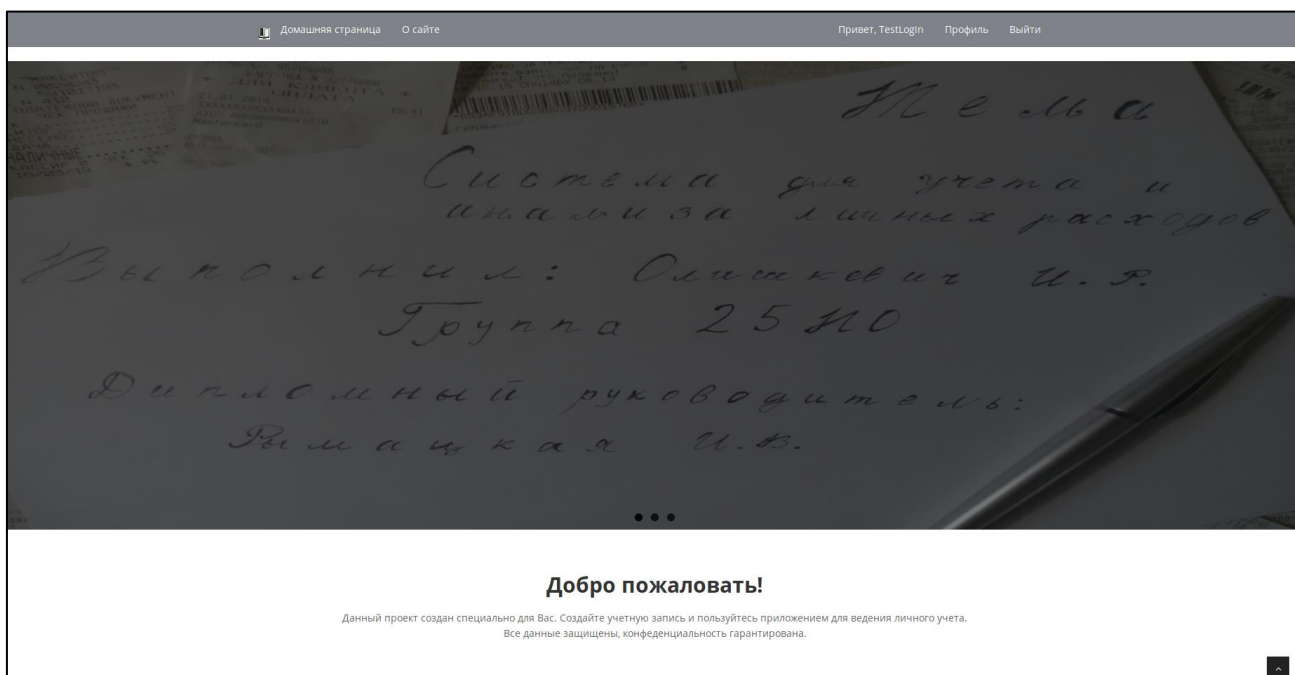


Рисунок Б.1 – Главная страница сайта

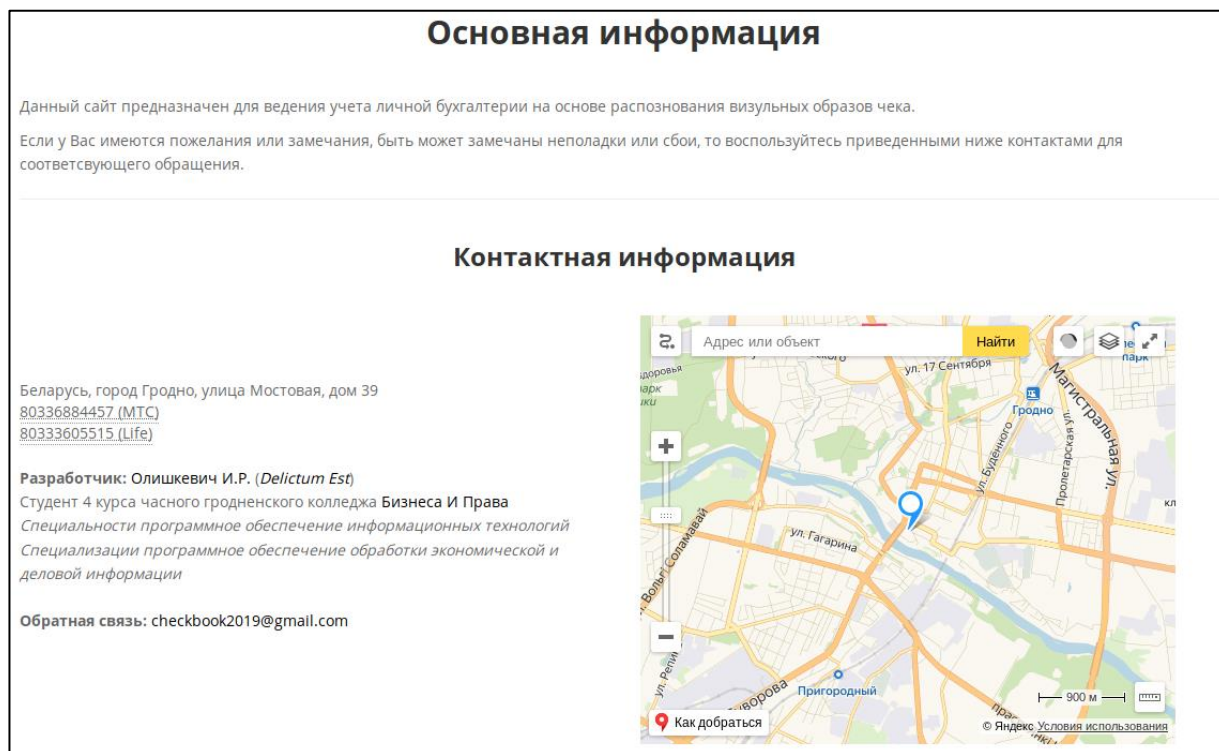


Рисунок Б.2 – Страница «О сайте»

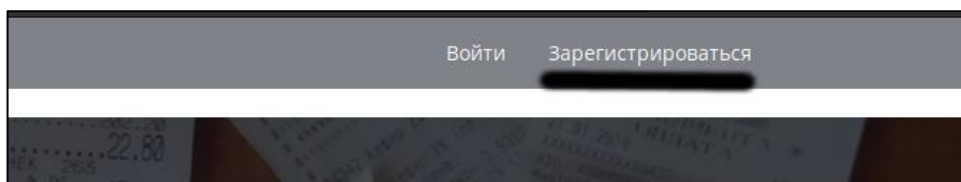


Рисунок Б.3 – Переход к регистрации

 A screenshot of a web page titled 'Регистрация' (Registration). The page has a header with 'Домашняя страница' (Home page) and 'О сайте' (About the site). The main content area contains four input fields labeled 'Логин' (Login), 'Email', 'Пароль' (Password), and 'Повторите пароль' (Repeat password). Below these fields is a 'Зарегистрироваться' (Register) button. At the bottom, there is a link: 'Уже зарегистрированы? Войти.' (Already registered? Login.).

Рисунок Б.4 – Отображение страницы регистрации

 A close-up screenshot of the registration form. The 'Логин' field contains 'TestLogin'. The 'Email' field contains 'eragon-i@tut.by'. The 'Пароль' and 'Повторите пароль' fields are filled with a series of dots, indicating masked text.

Рисунок Б.5 – Заполненные данные регистрации



Рисунок Б.6 – Присланное письмо подтверждения регистрации



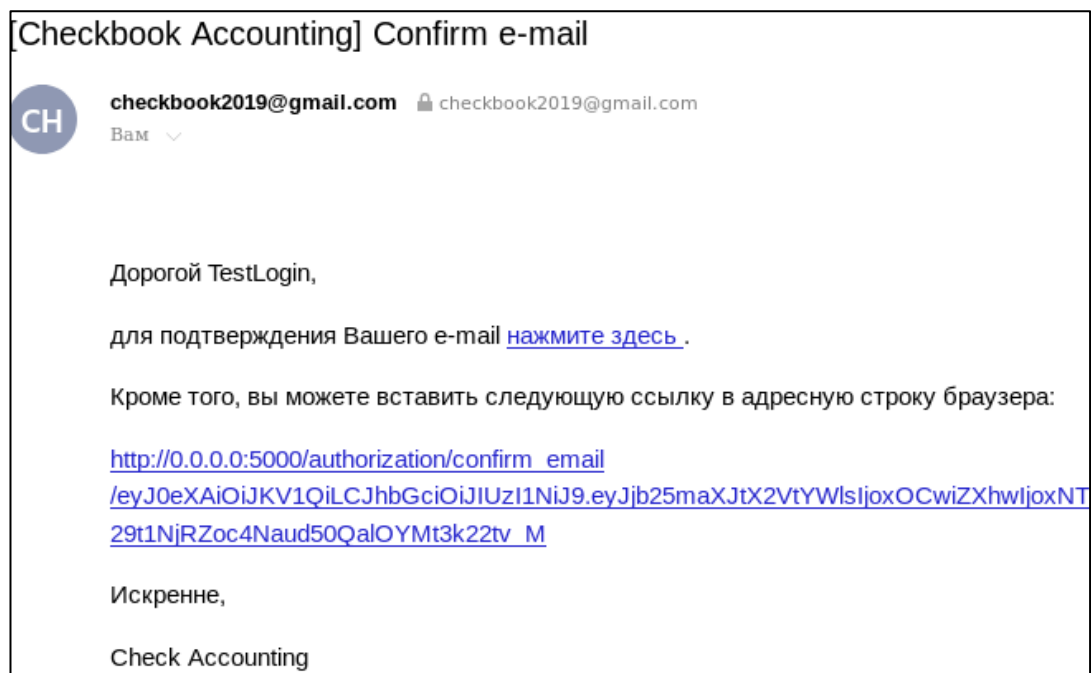


Рисунок Б.7 – Содержание письма с подтверждением регистрации

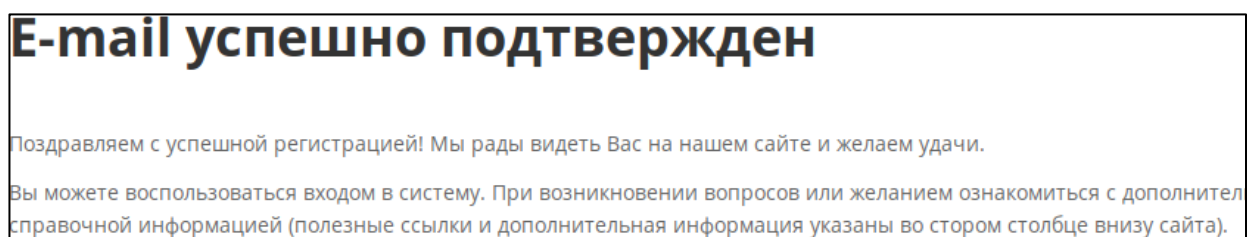


Рисунок Б.8 – Отображение подтвержденной учетной записи

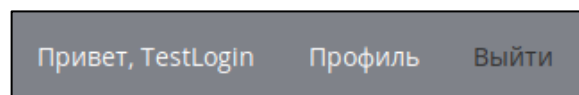


Рисунок Б.9 – Переход к выходу из учетной записи

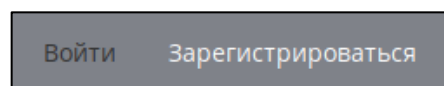


Рисунок Б.10 – Переход к авторизации

**Логин**

**Пароль**

☐ Запомнить меня

Рисунок Б.11 – Заполненные данные формы входа

The screenshot shows a web form titled "Сброс пароля" (Reset Password). It contains a label "Email" above a text input field. Below the input field is a button labeled "Сбросить пароль" (Reset Password).

Рисунок Б.12 – Отображение страницы сброса пароля

The screenshot shows an email interface. At the top, it says "[Check Accounting] Сброс пароля". Below this is a header area with a circular profile picture containing the letters "CH", the email address "checkbook2019@gmail.com", and a lock icon followed by "checkbook2019@gmail.com". Below the header, the email body contains the following text:

Дорогой TestLogin,

Для сброса пароля [нажмите здесь](#).

Кроме того, вы можете вставить следующую ссылку в адресную строку браузера:

[http://0.0.0.0:5000/authorization/reset\\_password/eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyZXNldF9wYXNzd29yZCI6MTgsImV4cCI6Ij7aEZ5iEEAuR1I](http://0.0.0.0:5000/authorization/reset_password/eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyZXNldF9wYXNzd29yZCI6MTgsImV4cCI6Ij7aEZ5iEEAuR1I)

Если вы не запрашивали сброс пароля, просто проигнорируйте это сообщение.

Искренне,

Check Accounting

Рисунок Б.13 – Письмо с подтверждением сброса пароля

The screenshot shows a web form titled "Сброс пароля" (Reset Password). It contains two labels: "Пароль" (Password) and "Повторите пароль" (Repeat password), each followed by a text input field. Below the input fields is a button labeled "Сбросить пароль" (Reset Password).

Рисунок Б.14 – Отображение страницы со сброшенным паролем

# Страница не найдена

То, что вы искали, просто не существует.

Попробуйте изменить адрес или сообщите о неработающем контенте.

[Вернуться на домашнюю страницу](#)


*P.S.*

*Мы надеемся, что Вы ничего не пытались взломать ^\_^*



Рисунок Б.15 – Отображение ошибки 404

[Домашняя страница](#) [О сайте](#) Привет, TestLogin [Профиль](#) [Выйти](#)



Зарботная плата: 0

Зарботок в день: 0

В этом месяце израсходовано: 0

Остаток в месяце: 0

Сегодняшние затраты: 0

Остаток на сегодня: 0.00

[Редактировать профиль](#)

[Подробная информация](#)

[Добавить чек](#)

Чеки отсутствуют

Рисунок Б.16 – Отображение страницы «Профиль»

Дата

25.05.2019

Организация

Добавить поле +

Название продукта	Цена	Продукт	Тип	Категория
<div>Хлеб Тракайский</div>	<div>12345.67</div>	<div></div>	<div></div>	<div>Строительные товары</div>

Внести

Загрузить

Обзор...

Очистить

Файлы не выбраны.

Обработать

Рисунок Б.17 – Страница добавления чека

# Редактирование профиля

Логин

TestLogin

Заработная плата

500

Отправить

Рисунок Б.18 – Заполненные данные страницы «Редактирование профиля»

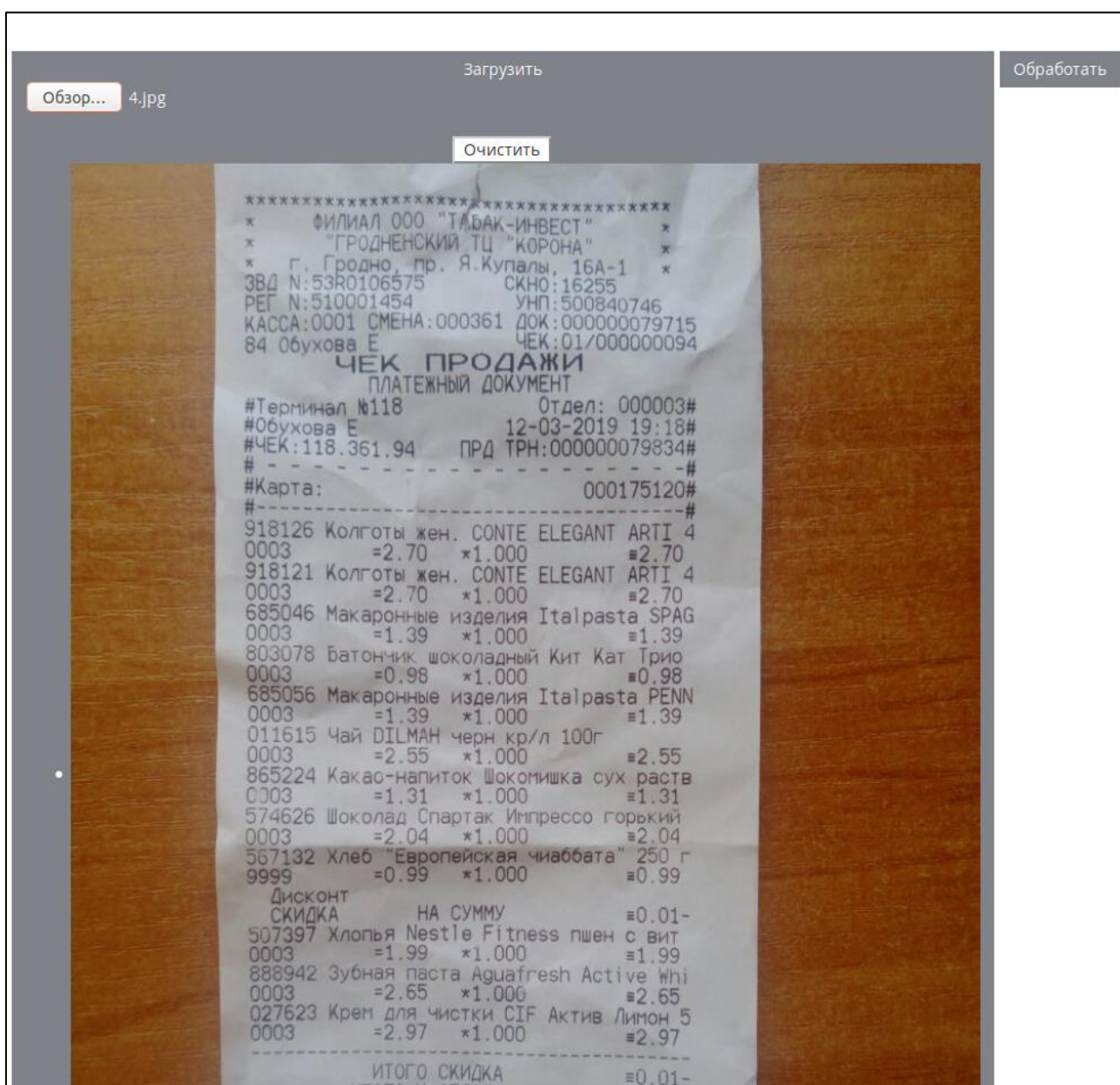



Рисунок Б.19 – Отображенный чек



Зарботная плата: 500.00

Зарботок в день: 16.67

В этом месяце израсходовано: 22.98

Остаток в месяце: 477.02

Сегодняшние затраты: 22.98

Остаток на сегодня: -6.31

Редактировать профиль

Подробная информация

Добавить чек

#

Чек от 2019-05-25 00:00:00

Удалить

Удалить выделенные

Рисунок Б.20 – Измененный профиль

12.03.2019

ООО I-ИНВЕСТ

\* г. Гродно, пр. Я.Купалы. 16А-1 «

500840746


Название продукта	Цена	Продукт	Тип	Категория
Удалить строку				
918126 Колготы жен. COF	2.70	Колготы	Колготки, легинсы	Одежда
Удалить строку				
918121 Колготы жен. COF	2.70	Колготы	Колготки, легинсы	Одежда
Удалить строку				
685046 Макароны изд	1.39	Макароны изделия	Мучные	Продовольственные продукты
Удалить строку				
803078 батон-** шоколад	0.98	Батон	Мучные	Продовольственные продукты
Удалить строку				
685056 Макароны изд	1.39	Макароны изделия	Мучные	Продовольственные продукты
Удалить строку				
011615 Чай DILMAN черн	2.55	Чай	Безалкогольные напитки	Продовольственные продукты
Удалить строку				
865224 Какао-напиток Ш	1.31	Кефир	Молочные	Продовольственные продукты
Удалить строку				
574626 Шоколад Спартак	2.04	Шоколад	Сладости	Продовольственные продукты

Рисунок Б.21 – Просмотр данных распознанного чека





### Check Accounting представляет



### Наши решения

- Visual check book # 1
- Repair planning # 2
- Encrypted dialogue # 3

Это windows-приложение позволяет вести диалог в зашифрованном виде.

Перейти

Его улучшенная версия в виде web-сайта использует принцип стеганографии посредством файлов.

Рисунок Б.25 – Центральное представление главной страницы

Открытие «app\_v.1.0.2.apk»

Вы собираетесь открыть:

**app\_v.1.0.2.apk**  
являющийся: Android package (28,4 МБ)  
из <http://0.0.0.0:5000>

Как Firefox следует обработать этот файл?

☒ Открыть в **OpenJDK Java 11 Runtime (по умолчанию)**

☐ Сохранить файл

☐ Выполнять автоматически для всех файлов данного типа.

Отмена ОК

#### Android-приложение

Что может быть лучше, чем загрузить чек не отходя от кассы? Именно для этого создано наше android-приложение.

Скачать

#### Защищенность данных

Мы ценим своих пользователей. Ознакомьтесь с **политикой конфиденциальности** и **пользовательским соглашением**

Ознакомиться

#### Руководство пользователя

Вы всегда можете обратиться к **справочнику**. Но если нужного ответа не найдется, напишите по адресу [checkbook2019@gmail.com](mailto:checkbook2019@gmail.com).

Перейти

Рисунок Б.26 – Диалог загрузки android-клиент

## Политика в отношении обработки персональных данных

- 1. Общие положения.**

Настоящая политика обработки персональных данных составлена в соответствии с требованиями Республиканского закона от 27.07.2006. №152-ФЗ «О персональных данных» и определяет порядок обработки персональных данных и меры по обеспечению безопасности персональных данных Олишкевича Игоря Руслановича (далее – Оператор). Оператор ставит своей важнейшей целью и условием осуществления своей деятельности соблюдение прав и свобод человека и гражданина при обработке его персональных данных, в том числе защиты прав на неприкосновенность частной жизни, личную и семейную тайну. Настоящая политика Оператора в отношении обработки персональных данных (далее – Политика) применяется ко всей информации, которую Оператор может получить от посетителя веб-сайта <https://mysite.ru>.
- 2. Основные понятия, используемые в Политике**

Автоматизированная обработка персональных данных – обработка персональных данных с помощью средств вычислительной техники; Блокирование персональных данных – временное прекращение обработки персональных данных (за исключением случаев, если обработка необходима для уточнения персональных данных); Веб-сайт – совокупность графических и информационных материалов, а также программ для ЭВМ и баз данных, обеспечивающих их доступность в сети интернет по сетевому адресу <https://mysite.ru>; Информационная система персональных данных — совокупность содержащихся в базах данных персональных данных, и обеспечивающих их обработку информационных технологий и технических средств; Обезличивание персональных данных — действия, в результате которых невозможно определить без использования дополнительной информации принадлежность персональных данных конкретному Пользователю или иному субъекту персональных данных; Обработка персональных данных – любое действие (операция) или совокупность действий (операций), совершаемых с использованием средств автоматизации или без использования таких средств с персональными данными, включая сбор, запись, систематизацию, накопление, хранение, уточнение (обновление, изменение), извлечение, использование, передачу (распространение, предоставление, доступ), обезличивание, блокирование, удаление, уничтожение персональных данных; Оператор – государственный орган, муниципальный орган, юридическое или физическое лицо, самостоятельно или совместно с другими лицами организующие и (или) осуществляющие обработку персональных данных, а также определяющие цели обработки персональных данных, состав персональных данных, подлежащих обработке, действия (операции), совершаемые с персональными данными; Персональные данные – любая информация, относящаяся прямо или косвенно к определенному или определяемому Пользователю веб-сайта <https://mysite.ru>; Пользователь – любой посетитель веб-сайта <https://mysite.ru>; Предоставление персональных данных – действия, направленные на раскрытие персональных данных определенному лицу или определенному кругу лиц; Распространение персональных данных – любые действия, направленные на раскрытие персональных данных неопределенному кругу лиц (передача персональных данных) или на ознакомление с персональными данными неограниченного круга лиц, в том числе обнародование персональных данных в средствах массовой информации, размещение в информационно-телекоммуникационных сетях или предоставление доступа к персональным данным каким-либо иным способом; Трансграничная передача персональных данных – передача персональных данных на территорию иностранного государства органу власти иностранного государства, иностранному физическому или иностранному юридическому лицу; Уничтожение персональных данных – любые действия, в результате которых персональные данные уничтожаются безвозвратно с невозможностью

Рисунок Б.27 – Представление политики конфиденциальности

# Пользовательское соглашение

Утверждено  
приказом управляющего ООО «Check Accounting»  
от 23.05.2019 № 17-ОД.

Вводится в действие с 01.12.2018.

## ПОЛЬЗОВАТЕЛЬСКОЕ СОГЛАШЕНИЕ

о порядке использования информационного электронного ресурса в сети интернет, размещенного по электронному адресу **Check Accounting**

Республика Беларусь, г. Гродно

### 1. Общие положения. Термины и определения. Предмет соглашения

1.1. В настоящем Пользовательском соглашении используются следующие термины и определения:

1.1.1. **Информационный электронный ресурс** — комплекс программно-технических средств и программных кодов, посредством которых Администрация предоставляет Пользователям информацию, обеспечивает доступ к соответствующим ресурсам и сервисам системы (в целях настоящего Пользовательского соглашения таким ресурсом является **Check Accounting**);

1.1.2. **Администрация ресурса** — собственник Информационного электронного ресурса, обеспечивающий функционирование Информационного электронного ресурса и доступ к нему, а именно ООО «Check Accounting»;

1.1.3. **Пользователь** — физическое лицо, присоединившееся к настоящему Пользовательскому соглашению и получившее доступ к информационным ресурсам и сервисам Информационного электронного ресурса, в том числе прошедшее процедуры регистрации и (или) идентификации на Информационном электронном ресурсе, а равно использующее его каким-либо образом без прохождения таких процедур, считается присоединившимся к настоящему Пользовательскому соглашению;

1.1.4. **Информация** — любая текстовая, графическая, звуковая, программно-техническая информация и файлы, размещаемая Пользователями на Информационном электронном ресурсе.

1.1.5. **Идентификация** — совокупность мероприятий по установлению и подтверждению достоверности сведений о пользователях Информационного электронного ресурса, условия и порядок которой предусмотрены настоящим Пользовательским соглашением.

1.1.6. **Учетная запись** — хранящаяся в компьютерной системе Информационного электронного ресурса совокупность данных о пользователе, указанных при регистрации и (или) идентификации пользователем, необходимая для его идентификации и предоставления доступа к его личным данным и настройкам.

1.1.7. **ID номер** — идентификационный номер, который присваивается учетной записи Пользователя после прохождения процедуры регистрации на Информационном электронном ресурсе.

1.1.8. **Формы Пользовательского соглашения** — это формы, в которые Пользователь вносит информацию о своих персональных данных при прохождении процедур регистрации и (или) идентификации, являющиеся неотъемлемой частью настоящего Пользовательского соглашения.

## Рисунок Б.28 – Представление пользовательского соглашения

### Авторизация

- Регистрация
- Вход в учетную запись
- Выход из учетной записи

*Предполагается, что пользователь уже вошел в учетную запись системы (см. пункт "Вход в учетную запись").*

Нажмите "Выйти" в правой части навигационного меню (оно расположено в верхней части сайта), как показано на рисунке 3.1, или добавьте в адресную строку сразу после названия сайта следующее: "/authorization/logout".

Привет, TestLogin    Профиль    Выйти

Рисунок 3.1 - Переход к выходу из учетной записи

Вы должны были успешно выйти из системы и оказаться на главной странице. Если это так, в навигационном меню в правой части теперь должно отображаться "Войти" и "Зарегистрироваться", как видно на картинке 2.1.

- Сброс пароля

### Андроид

- Первый запуск
- Регистрация
- Вход в учетную запись
- Выход из учетной записи
- Загрузка чека
- Настройки
- Помощь

## Рисунок Б.29 – Представление справочной системы



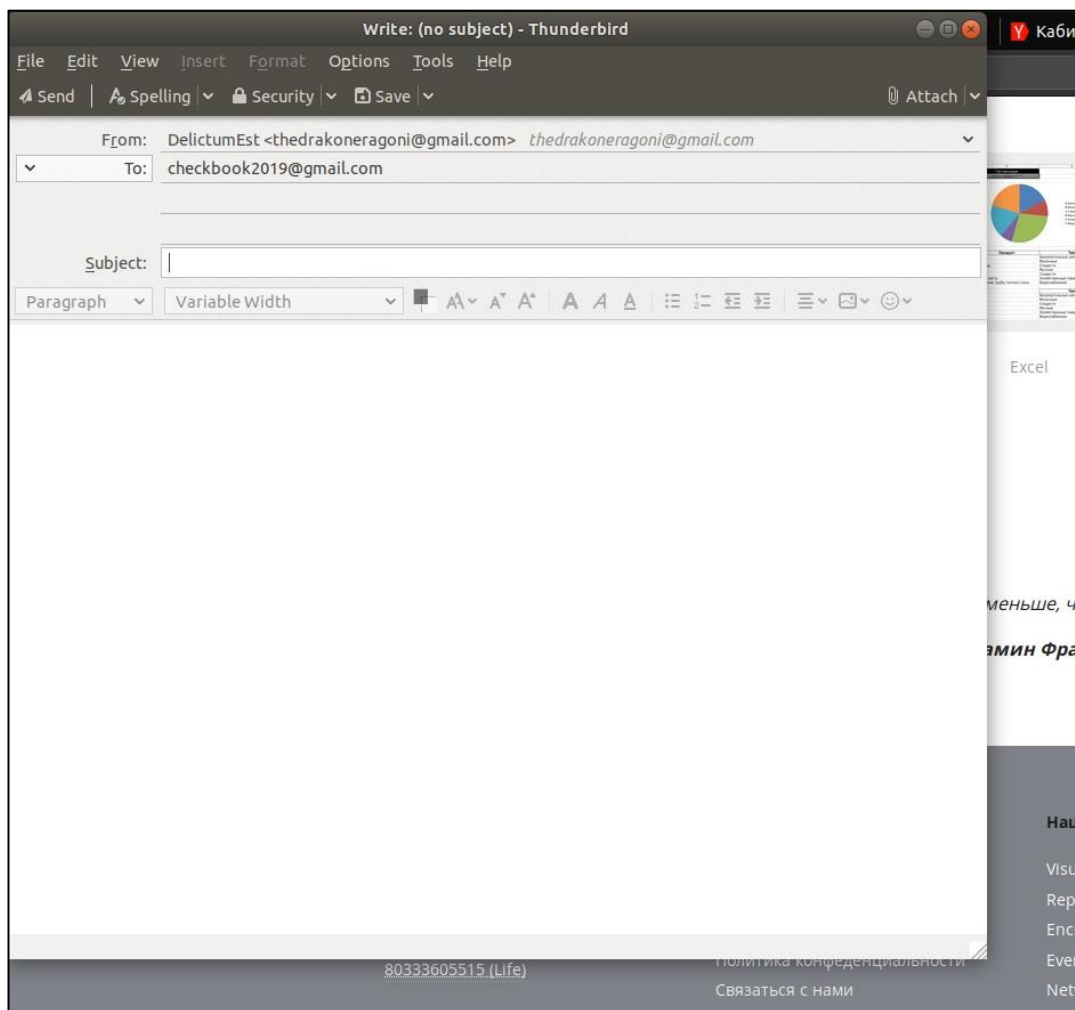


Рисунок Б.30 – Открытие приложения для отправки писем

2019-03-12

ООО I-ИНВЕСТ""""

Название продукта	Цена	Продукт	Категория	Тип
011615 Чай DILMAN черн кр/л 100г	2.55	Чай	Продовольственные продукты	Безалкогольные напитки
865224 Какао-напиток Шокомишка сух раств	1.31	Кефир	Продовольственные продукты	Молочные
574626 Шоколад Спартак Иппрессо горький	2.04	Шоколад	Продовольственные продукты	Сладости
567132 Хлеб "Европейская чизббата" 250 г	1.00	Хлеб	Продовольственные продукты	Мучные
507397 Хлопья Nestle Fitness пшен с вит	1.99	Хлопья	Продовольственные продукты	Сладости
888942 Зубная паста Muuafresn Active Whi	2.65	Зубная паста	Бытовые товары	Хозяйственные товары
027623 Креп для чистки CIF Актив Липон 5	2.97	Крепление трубы теплого пола	Строительные товары	Водоснабжение
Итого: 14.51				

Eagle Image 3 of 5


Рисунок Б.31 – Раскрытая фотография в диалоговом режиме

#		
Чек от 2019-06-03 15:11:44	Удалить	<input type="checkbox"/>
Чек от 2019-06-02 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-06-01 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-30 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-30 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-29 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-27 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-25 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-05-22 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2019-03-28 00:00:00	Удалить	<input type="checkbox"/>
Старые чеки		Удалить выделенные

Рисунок Б.32 – Первая страница таблицы

#		
Чек от 2019-03-12 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2018-05-27 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2018-03-16 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2013-07-24 00:00:00	Удалить	<input type="checkbox"/>
Чек от 2013-05-28 12:23:00	Удалить	<input type="checkbox"/>
Чек от 2013-05-28 12:23:00	Удалить	<input type="checkbox"/>
Чек от 2007-03-14 15:44:00	Удалить	<input type="checkbox"/>
Новые чеки		Удалить выделенные


Рисунок Б.33 – Последняя страница таблицы



## Бухгалтерский учет

Используя данную систему Вы сможете  
лучше контролировать свое  
финансовое положение и  
способствовать его улучшению.

[В профиль](#)



## Android- приложение

Что может быть лучше, чем загрузить  
чек не отходя от кассы? Именно для  
этого создано наше android-  
приложение.

[Скачать](#)

## Защи д

Мы ценим  
Ознако  
конфе  
пользовате

[Оз](#)

Рисунок Б.34 – Раздел ознакомления в центральной части главной страницы

Дата: 25.09.2017

Организация

Добавить поле +

Название продукта

Что-то

Календарь: сентябрь 2017 г.

пн	вт	ср	чт	пт	сб	вс
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

Рисунок Б.35 – Выбор даты

25.05.2019

Sandbox's

Sunshine

500840123

Название продукта	Цена	Продукт	Тип	Категория
Хлеб "Молочный"	0.98	Хлеб	Мучные	Продовольственные продукты
Восточное вино	6.00	Вино	Алкогольные напитки	Продовольственные продукты
Пена монтажная	16.00	Пена монтажная	Пены	Строительные товары

Общая стоимость чека: 22.98

Рисунок Б.36 – Добавленный чек ручным вводом

## ПРИЛОЖЕНИЕ В

(справочное)

### Диаграммы и схемы

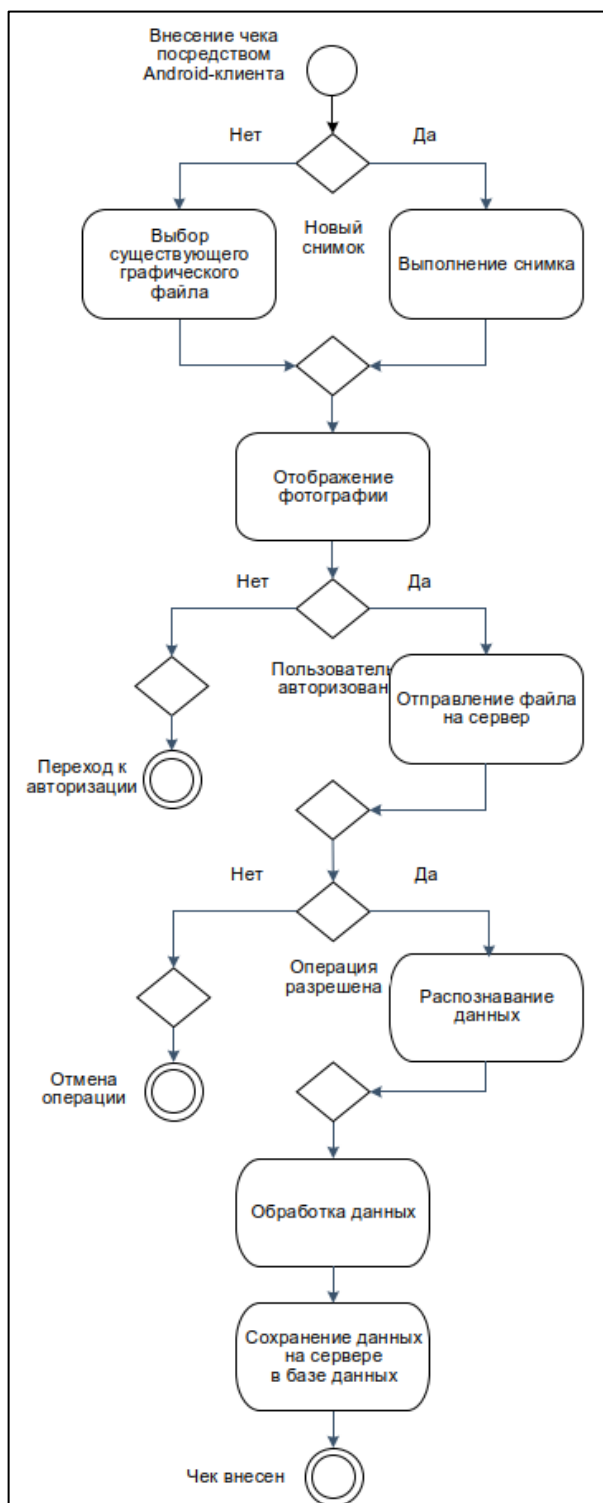


Рисунок В.1 – Диаграмма деятельности

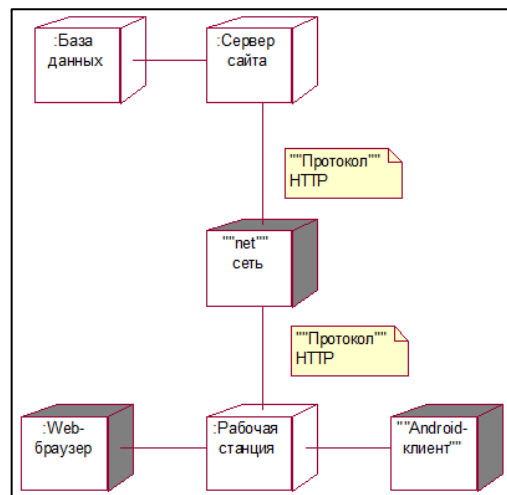


Рисунок В.2 – Диаграмма развертывания



Рисунок В.3 – Схема «Сущность-связь»

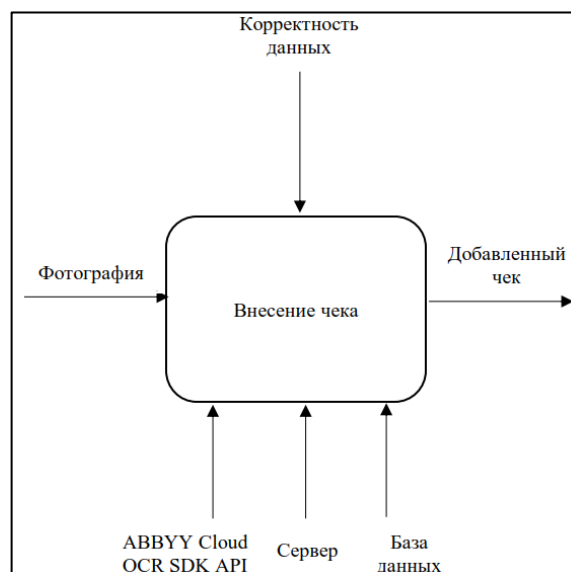


Рисунок В.4 – Схема «IDEF-0»

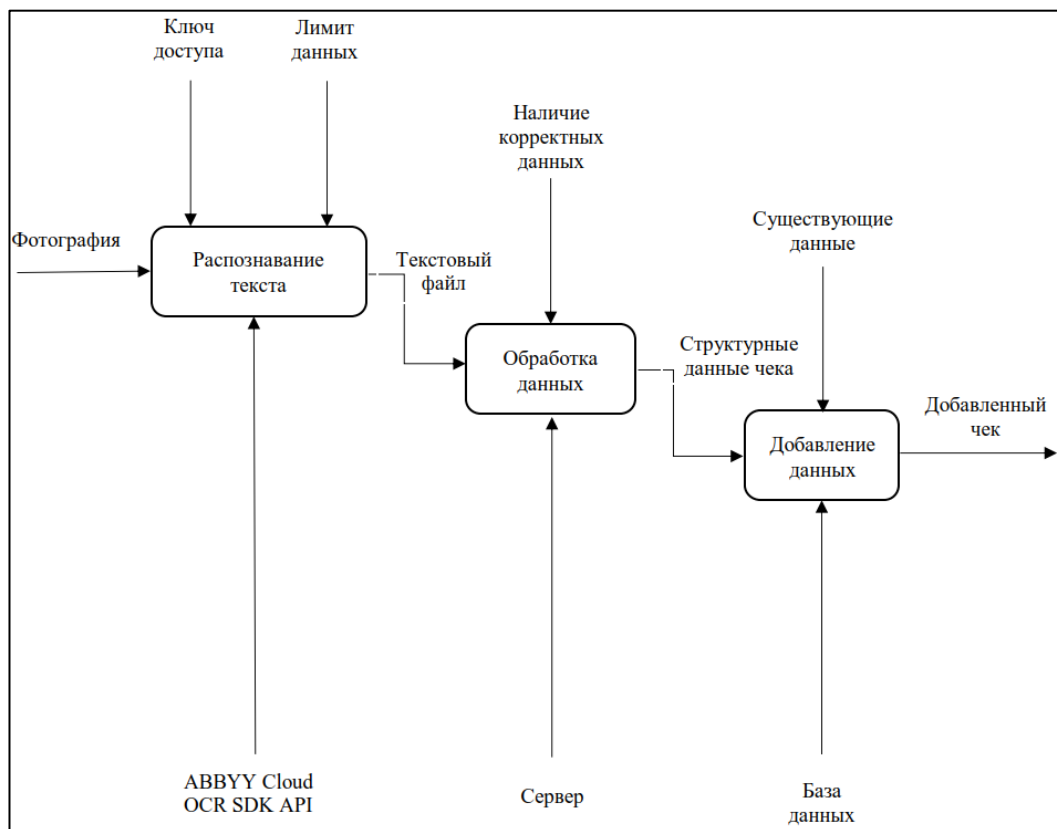


Рисунок В.5 – Схема «IDEF-0» – декомпозиция

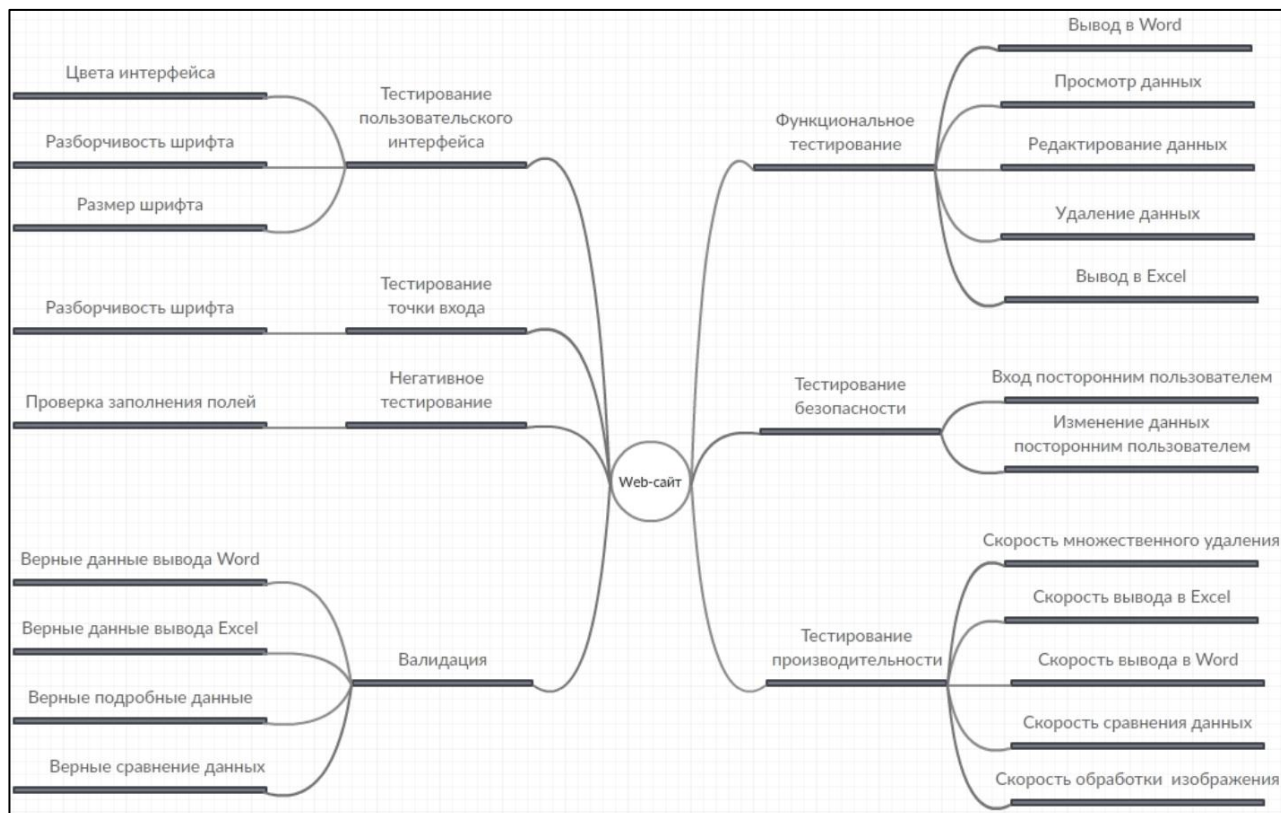


Рисунок В.6 – Схема тестирования программного средства

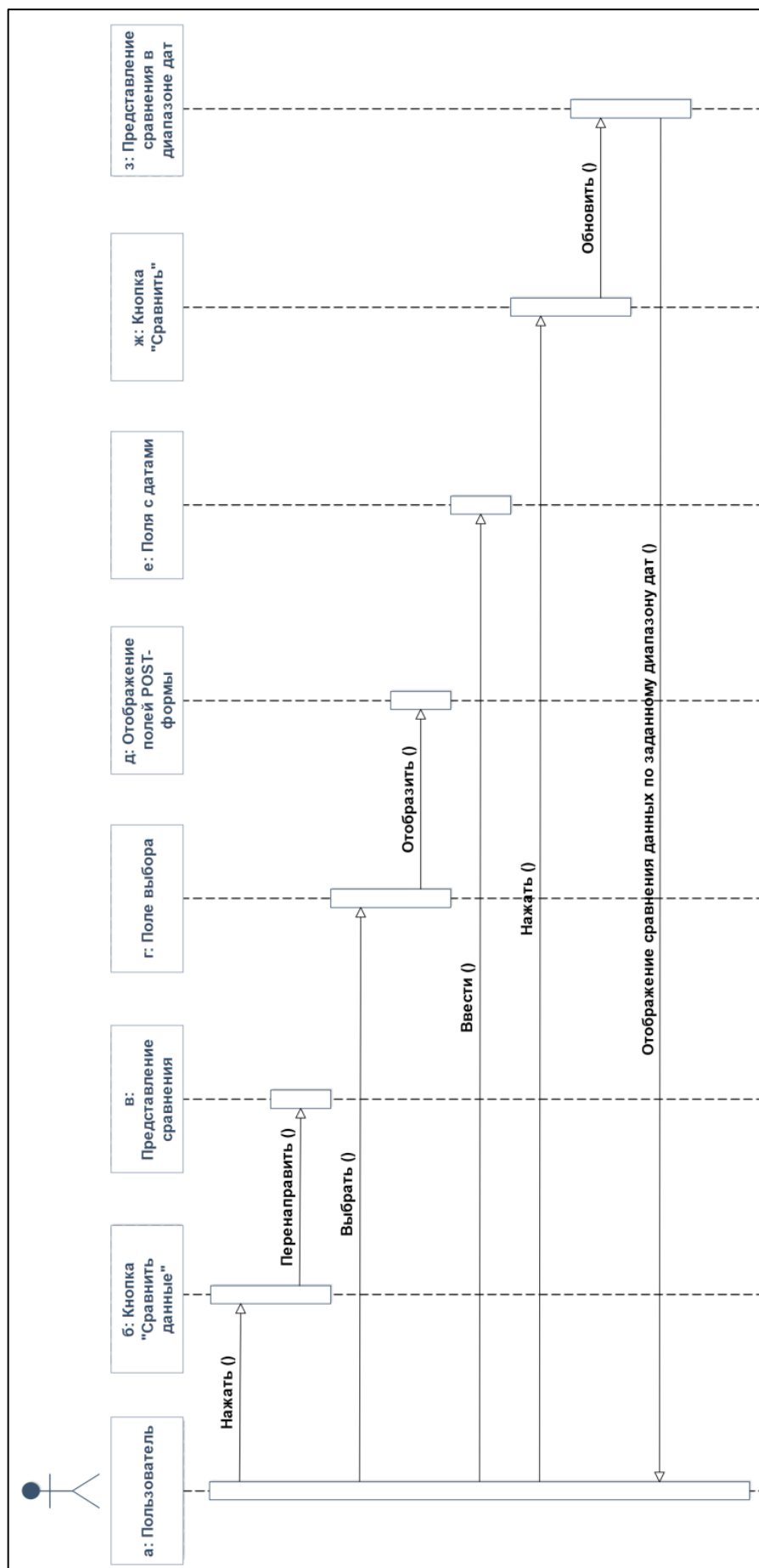


Рисунок В.7 – Диаграмма последовательности действий

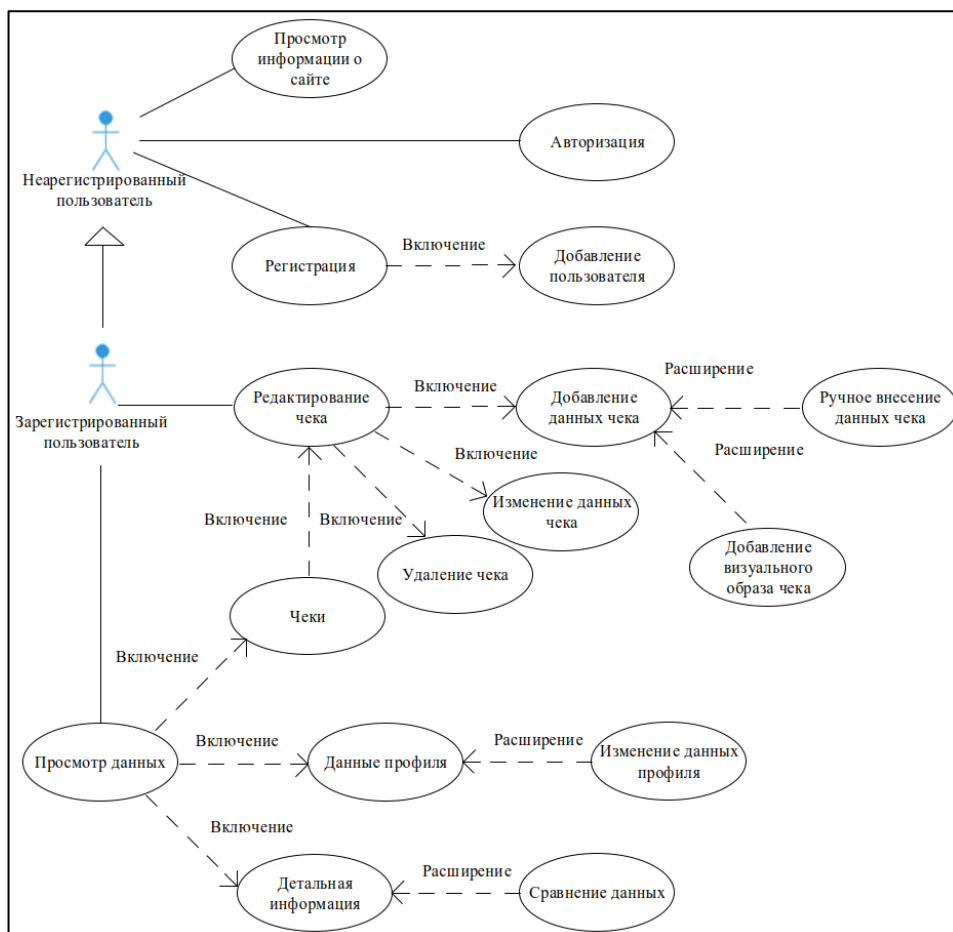


Рисунок В.8 – Диаграмма вариантов использования

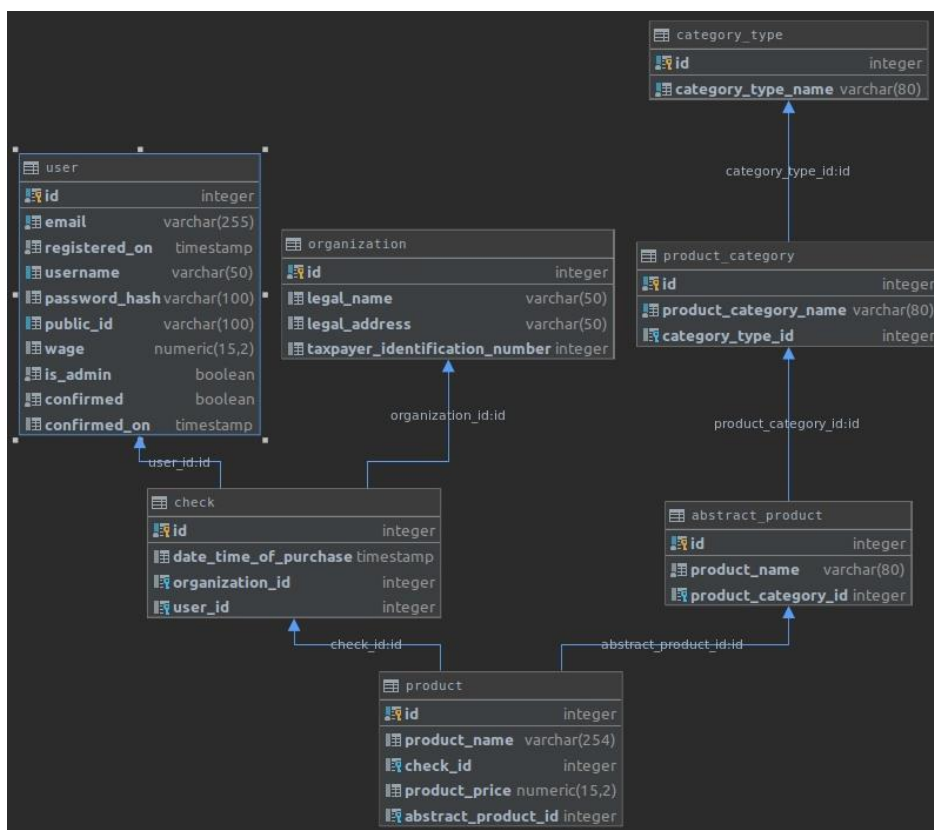


Рисунок В.9 – Реляционная модель данных



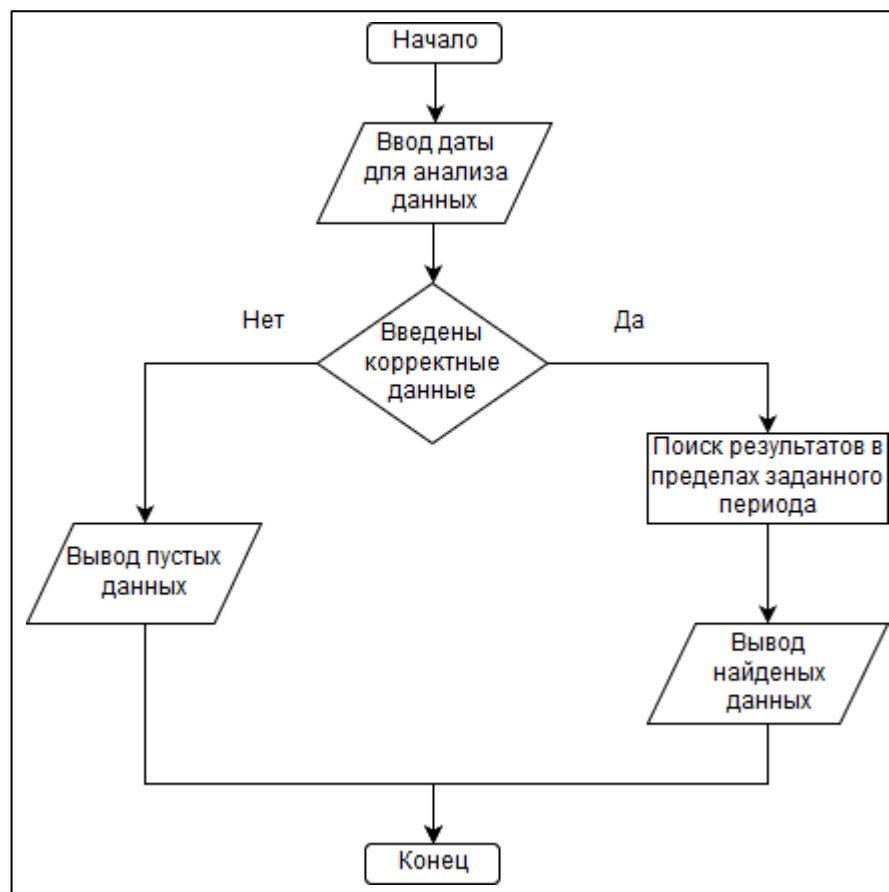


Рисунок В.10 – Блок-схема