

1 ОЗНАКОМЛЕНИЕ С ВЫЧИСЛИТЕЛЬНЫМ ЦЕНТРОМ ПРЕДПРИЯТИЯ

В отделах организации «Devcraft» осуществляют работу множество компьютеров, связанных между собой одной сетью. Благодаря этой сети сотрудники предприятия могут обмениваться информацией между отделами и следить за изменениями, внесенными другими сотрудниками в общую базу данных.

Схема подключения компьютеров по локальной сети, отображена на рисунке 1.



Рисунок 1 – Схема подключения компьютеров по локальной сети

Отдел разработки и тестирования программного обеспечения занимается планированием структуры приложения, его разработкой и тестированием с последующим устранением ошибок.

Коммерческий отдел выстраивает отношения с клиентами, а также ведет отчетность, на основе которой проводится аналитика продуктивности работы предприятия.

На данных компьютерах установлена операционная система Linux семейства Ubuntu 18 LTS. Работа осуществляется в следующих интеграционных системах разработки:

- 1) WebStorm – мощная IDE для веб-разработки на JavaScript. Отличается удобным и умным редактором JavaScript, HTML и CSS и поддержкой новых технологий и языков, таких как TypeScript, CoffeeScript, Dart, Less, Sass и Stylus.;
- 2) PhpStorm – это умная IDE для языка PHP и других веб-технологий, понимающая код и отличающаяся интеллектуальным редактором, автодополнением кода, рефакторингами, встроенным отладчиком и другими инструментами;
- 3) Rider – кроссплатформенная IDE для платформы .NET, построенная на базе IntelliJ IDEA и ReSharper. Поддерживает C#, VB.NET, ASP.NET, XAML, XML, JavaScript, TypeScript, JSON, HTML, CSS и SQL.;
- 4) Visual Studio Code – редактор исходного кода, разработанный Microsoft, для кроссплатформенной разработки веб- и облачных приложений.

Выше были перечислены основные программные средства. Также используется множество вспомогательного программного обеспечения, начиная web-браузерами и заканчивая плагинами для интеграционных систем разработок.

2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА

2.1 Введение

Техническое задание составлено для программы, предназначенной для учета личной бухгалтерии на основе распознавания визуальных образов чека.

2.2 Основание для разработки

Основанием для разработки программы, предназначенной для учета личной бухгалтерии на основе распознавания визуальных образов чека, является индивидуальное задание по преддипломной практике, утвержденное 01.04.2019.

2.3 Назначение разработки

Программа для учета личной бухгалтерии на основе распознавания визуальных образов чека предназначена для ведения расходов пользователя, отслеживании трат и сметы.

2.4 Требования к программному средству

2.4.1 Требования к функциональным характеристикам

При разработке данного сервиса необходимо реализовать удобный и интуитивно понятный интерфейс. Кроме того, должна быть реализована справочная система для того, чтобы узнать о функциях программы, и разработчике.

Программа должна содержать следующие функции:

- 1) организация авторизации;

- 2) получение чека через api сервера;
- 3) обработка полученного изображения;
- 4) возможность просмотра добавленной информации, включая сведения о продавце, дате совершения сделки и непосредственных данных затрат.

Таким образом были перечислены основные функции программы.

2.4.2 Требования к надежности

Причиной нарушения работоспособности программ при безотказности аппаратуры всегда является конфликт между реальными исходными данными, подлежащими обработке, и программой, осуществляющей эту обработку. Работоспособность программного средства можно гарантировать при исходных данных, которые использовались при отладке и испытаниях. Реальные исходные данные могут иметь значения, отличающиеся от заданных техническим заданием и от использованных при тестировании программ. При таких исходных данных функционирование программ трудно предсказать заранее и весьма вероятны различные аномалии, завершающиеся отказами. В программе должны быть предусмотрены проверки функций во избежание получения не той информации, что необходима.

2.4.3 Требования к составу и параметрам технических средств

Для нормальной работы программы по учету личной бухгалтерии на основе распознавания визуальных образов чека необходимо устройство с выходом в Интернет. Кроме того, необходимо наличие любого браузера.

2.4.4 Требования к информационной и программной совместимости

Данный сайт был разработан в операционной системе Linux на дистрибутиве Ubuntu 18.04 и в среде разработки PyCharm Professional Edition

2019.1. Для правильной и корректной работы сайта необходима любая операционная система и современный браузер.

2.5 Перечень стадий, этапов и сроков разработки программы

При разработке программы для учета личной бухгалтерии на основе распознавания визуальных образов чека были выделены следующие основные стадии разработки программного средства:

- 1) формирование требований к программному средству (дата окончания разработки стадии: 05.04.2018);
- 2) проектирование (дата окончания разработки стадии: 19.04.2019);
- 3) реализация (дата окончания разработки стадии: 22.04.2019);
- 4) тестирование (дата окончания разработки стадии: 24.04.2019);
- 5) внедрение (дата окончания разработки стадии: 27.04.2019).

Основными этапами разработки программы являются:

- 1) «Постановка задачи» – это описание задачи по определенным правилам, которое даст исчерпывающее представление её сущности, логике преобразования информации для получения результата. На основе постановки задачи программист должен представить логику ее решения и рекомендовать стандартные программные средства, пригодные для ее реализации. Во время постановки задачи четко формулируется назначение разрабатываемого программного обеспечения и определяется список основных требований к нему. Каждое требование, по сути, есть описание необходимого заказчику свойства программного обеспечения. Этап постановки задачи заканчивается принятием основных проектных решений и разработкой технического задания, фиксирующего принципиальные требования к разрабатываемому программному обеспечению;
- 2) «Разработка пользовательского интерфейса» создание внешнего вида для клиентской части программного средства;

- 3) «Разработка программы» – представляет собой процесс поэтапного написания кода программы на выбранном языке программирования (кодирование), их тестирование и отладку;
- 4) «Отладка» – это процесс поиска и устранения ошибок. Процесс отладки выполняется по следующему алгоритму: после того как написан рабочий код производятся тестовые запуски программы на различных наборах тестовых данных. При этом тестер или программист заранее должны получить контрольный результат, с которым будет идти сверка работы проверяемого кода;
- 5) «Внедрение» – начинается после отладки программы. Происходит процесс развертывания в рабочем окружении и интеграция с android-клиентом.

2.6 Порядок контроля и приемки

Для того, чтобы воспользоваться web-приложением нужно написать адрес страницы в адресную строку любого браузера. На главной странице отображается информация общего характера по ознакомлению с программным средством.

При нажатии на кнопку «О сайте» появляется дополнительное представление (страница), на котором отображена детальная информация о данном проекте и разработчике.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

3.1 Разработка диаграмм языка UML

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования. UML призван поддерживать процесс моделирования программного средства на основе объектно-ориентированного подхода, 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. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

UML обеспечивает:

- 1) иерархическое описание сложной системы путем выделения пакетов;
- 2) формализацию функциональных требований к системе с помощью аппарата вариантов использования;
- 3) детализацию требований к системе путем построения диаграмм деятельности и сценариев;
- 4) выделение классов данных и построение концептуальной модели данных в виде диаграмм классов;
- 5) выделение классов, описывающих пользовательский интерфейс, и создание схемы навигации экранов;
- 6) описание процессов взаимодействия объектов при выполнении системных функций;
- 7) описание поведения объектов в виде диаграмм деятельности и состояний;
- 8) описание программных компонент и их взаимодействия через

интерфейсы;

9) описание физической архитектуры системы.

В UML предусмотрены следующие диаграммы:

- 1) диаграммы, описывающие поведение системы:
 - а) диаграммы состояний;
 - б) диаграммы деятельности;
 - в) диаграммы объектов;
 - г) диаграммы последовательностей;
 - д) диаграммы взаимодействия.
- 2) диаграммы, описывающие физическую реализацию системы:
 - а) диаграммы компонент;
 - б) диаграммы развертывания.

В данном проекте используются следующие виды диаграмм:

- 1) диаграмма деятельности;
- 2) диаграмма вариантов использования;
- 3) диаграмма последовательности.

Диаграмма деятельности – методология объектно-ориентированного особенностей проектирования, предназначенная для детализации алгоритмической и логической организации системы. Каждое действие расчленяется на фундаментальные процессы. Диаграммы деятельности используются независимо для визуализации, специфицирования, конструирования и документирования динамики совокупности объектов. В составе: для моделирования потока управления при выполнении некоторой операции. Диаграмма деятельности для данного проекта отображена на рисунке А.1.

Диаграмма последовательности – это упорядоченная по времени диаграмма взаимодействия, которую следует читать сверху вниз. У каждого варианта использования имеются альтернативные потоки. Каждая диаграмма описывает один из потоков варианта использования. Данная диаграмма, разработанная для программы по учету личной бухгалтерии на основе

распознавания визуальных образов чека, изображена на рисунке А.2.

Диаграмма вариантов использования это – описание последовательности действий, которые система осуществляет в ответ на внешние воздействия пользователей или других программных систем. Варианты использования отражают функциональность системы с точки зрения получения значимого результата для пользователя. Варианты использования предназначены в первую очередь для определения функциональных требований к системе и управляют всем процессом разработки. Диаграмма вариантов использования для данного проекта представлена на рисунке А.3.

3.2 Концептуальный прототип

Программное средство для учета личной бухгалтерии на основе распознавания визуальных образов чека будет представлять собой веб-приложение и android-клиент. По нажатию кнопки «+», предлагается выбор добавление фотографии из существующих или создание новой. После размещения визуального объекта в контейнер, разрешено использовать отправку на сервер посредством api, где требуется быть предварительно авторизованным, для дальнейшего распознавания и обработки.

Сервер, получая закодированную строку изображения, обрабатывает ее, ищет аналоги среди существующих, чтобы категорировать предметы оплаты. Результаты вышеописанного действия помещаются в базу данных для данного пользователя.

4 РАБОЧИЙ ПРОЕКТ

4.1 Функции и элементы управления

В качестве элементов управления на сайте для учета личной бухгалтерии на основе распознавания визуальных образов чека используются кнопки, контейнеры, меню и слайдинги.

4.2 Текст программы

Текст программного кода программы для учета личной бухгалтерии на основе распознавания визуальных образов чека представлен в приложении В.

5 ИСПЫТАНИЕ ПРОГРАММЫ

5.1 Функциональное тестирование

При запуске появилась главная страница web-клиента, изображенная на рисунке 1.

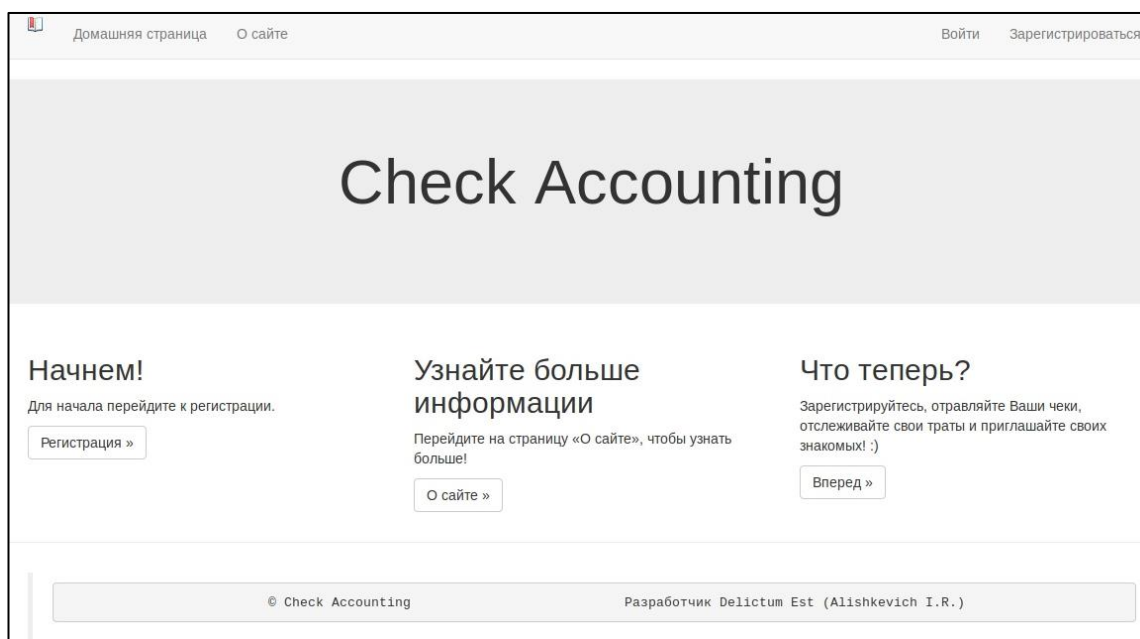


Рисунок 1 – Домашняя страница web-клиента

В навигационной панели было нажато на «О сайте», после чего произошло перенаправление на представление, отображенное на рисунке 2.

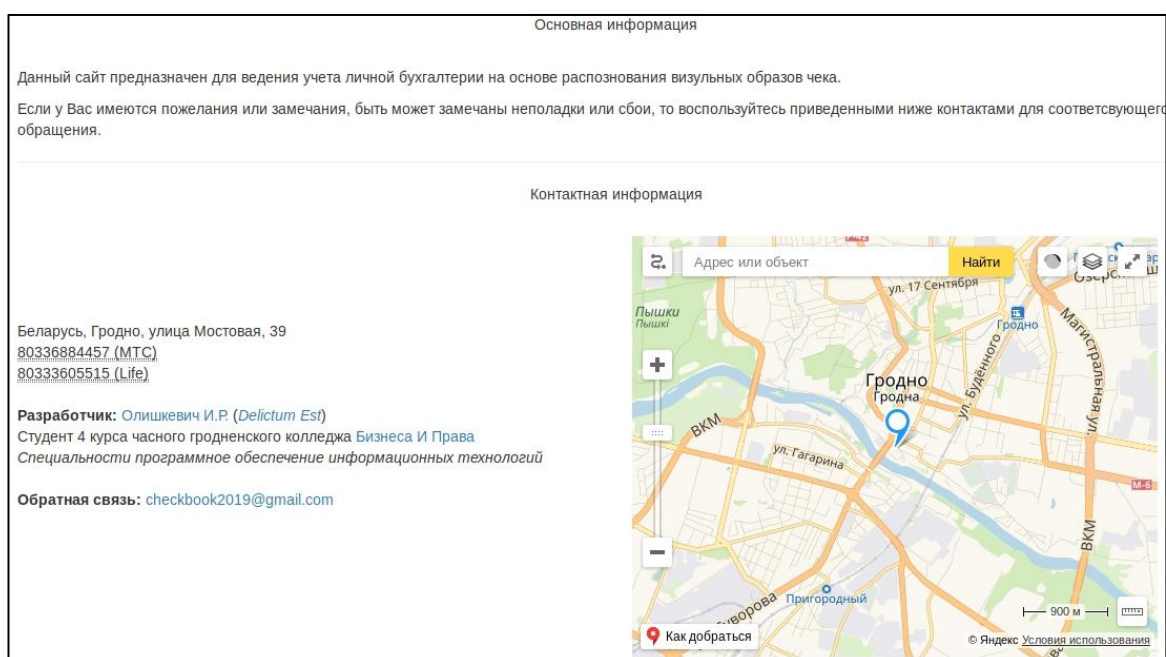
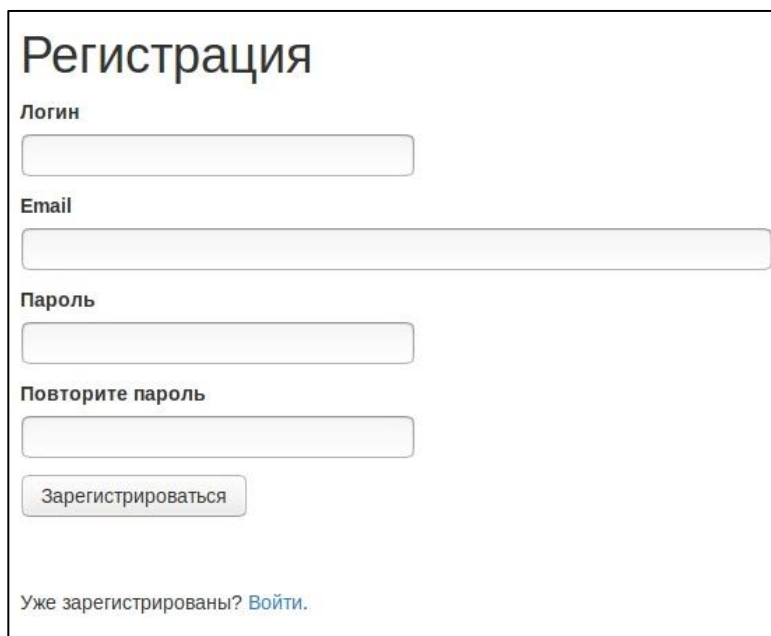


Рисунок 2 – Представление «О сайте»

Там же было нажато на «Домашняя страница», после чего произошло перенаправление к начальному представлению, указанному на рисунке 1.

Выбрано «Зарегистрироваться» в навигационном меню, после чего возникла следующая страница, отображенная на рисунке 3.

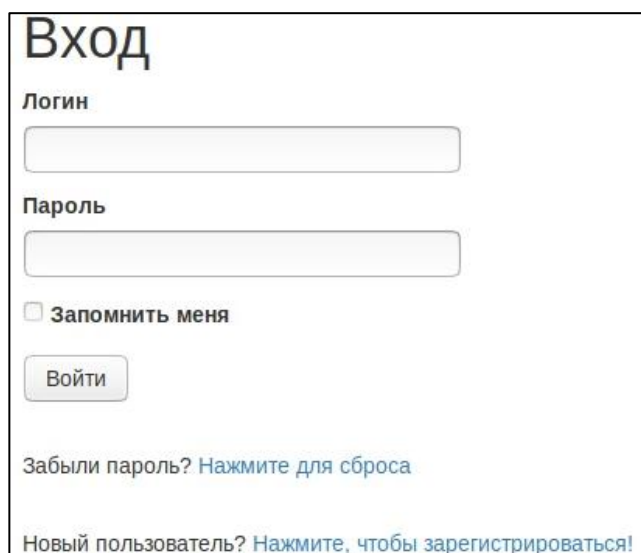


The screenshot shows a registration form with the title "Регистрация" in a large, bold font. Below the title are four input fields: "Логин" (Login), "Email", "Пароль" (Password), and "Повторите пароль" (Repeat password). Each field is represented by a light gray rectangular box. Below the "Повторите пароль" field is a button labeled "Зарегистрироваться" (Register). At the bottom of the form, there is a link that says "Уже зарегистрированы? Войти." (Already registered? Log in.).

Рисунок 3 – Страница «Регистрация»

Были заполнены данные и нажата кнопка «зарегистрироваться». На указанный почтовый адрес пришло письмо с подтверждением регистрации, где осуществлен переход по ссылке.

В навигационном меню выбрано «Войти», после чего открылась страница, как видно из рисунка 4.



The screenshot shows a login form with the title "Вход" in a large, bold font. Below the title are two input fields: "Логин" (Login) and "Пароль" (Password). Each field is represented by a light gray rectangular box. Below the "Пароль" field is a checkbox labeled "Запомнить меня" (Remember me). Below the checkbox is a button labeled "Войти" (Log in). At the bottom of the form, there are two links: "Забыли пароль? Нажмите для сброса" (Forgot password? Click to reset) and "Новый пользователь? Нажмите, чтобы зарегистрироваться!" (New user? Click to register!).

Рисунок 4 – Авторизация

Было нажато «Редактировать профиль», после чего произошло перенаправление на соответствующую страницу, как видно на изображении 5.

Домашняя страница О сайте Привет, Delictum Профиль Выйти

Редактирование профиля

Логин

Зарботная плата

Отправить

© Check Accounting Разработчик Delictum Est (Alishkevich I.R.)

Рисунок 5 – Редактирование профиля

Просмотрен один из существующих чеков, как это представлено на рисунке 6.

Домашняя страница О сайте Привет, Delictum Профиль Выйти

30.04.2019

Неопределено

None

None

1	Хлеба	Хлеб	1.00
2	Молоко	Молоко	1.19
3	Брюки джинсовые темные	Брюки	40.00

Общая стоимость чека: 42.19

© Check Accounting Разработчик Delictum Est (Alishkevich I.R.)

Рисунок 6 – Просмотр чека

В конечном итоге, можно утверждать, что функциональное тестирование было успешно пройдено и выявление ошибочных данных не было обнаружено.

5.2 Полное тестирование

В качестве полного тестирования произведено ручное добавление чека.

Произведен запуск сервера из IDE PyCharm, использующий входную точку «checkbook_accounting». Было удостоверено, что он находится во

включенном и рабочем состоянии.

Был осуществлен запуск сайта из браузера «Firefox Quantum». Результат запуска аналогичен изображению на рисунке 1.

Была нажата надпись в главном меню «Войти», после чего произошло перенаправление на страницу авторизации. Итог идентичен функциональному тестированию и представлен на рисунке 4.

Произведено заполнение данных: поле «Логин» – «thedrakoneragoni@gmail.com», поле «Пароль» – «keb1wagenfo7ack72».

После нажатия пункта «Профиль» в навигационном меню произошло перенаправление на соответствующую страницу, которая представлена на рисунке 7.

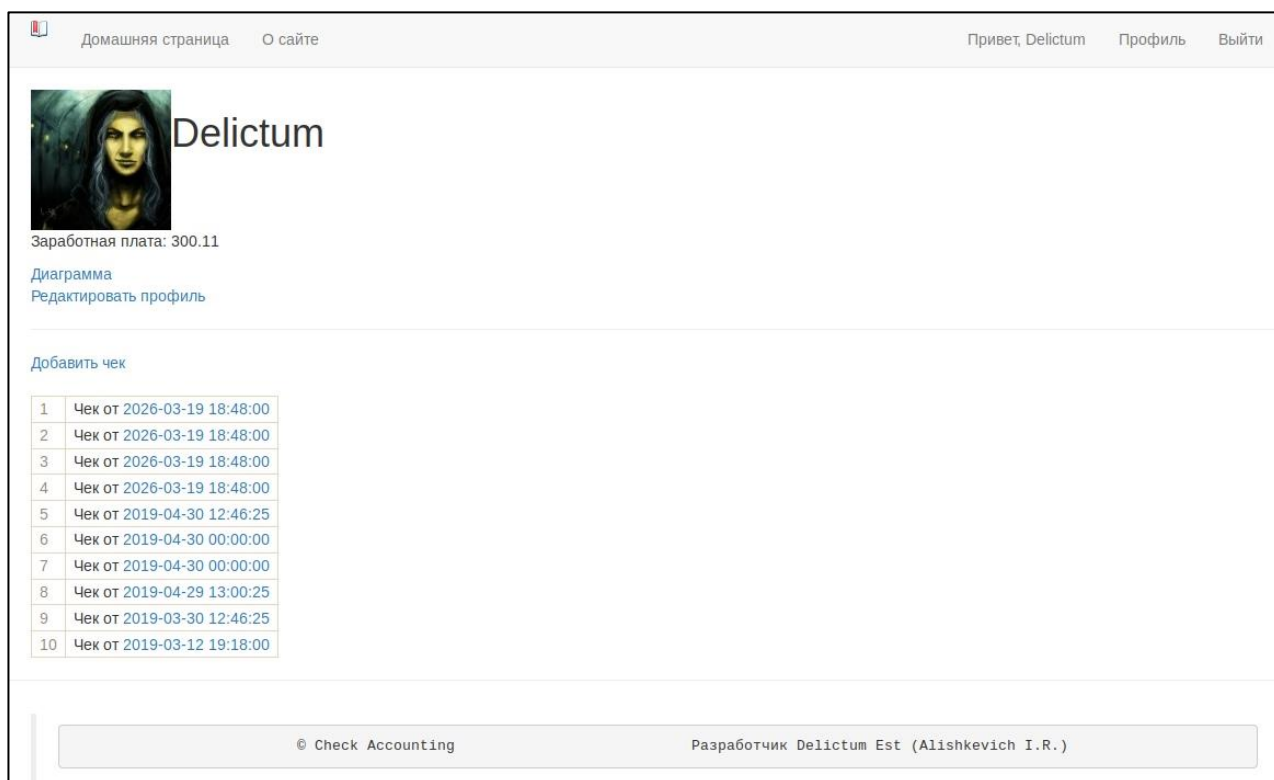


Рисунок 7 – Добавление чека

Далее была нажата надпись: «Добавить чек», которая отображена в центре изображения 7.

Выполнен ввод данных «Дата», где отобразился календарь. Далее было заполнено: «Название продукта» при вводе которого автоматически проставились категория продукта, его тип и наименование. Нажали «Добавить поле» и заполнили также другие четыре строки подобным образом, как

представлено на рисунке 8.

Дата: 30.04.2019

Организация:

Добавить поле +

Название продукта	Цена	Продукт		
Хлеба	12345.67	Хлеб	Мучные	Продовольственные продукты
Удалить строку				
Мол	12345.67	Молоко	Молочные	Продовольственные продукты
Удалить строку				
Хлебцы	12345.67	Хлебцы	Мучные	Продовольственные продукты
Удалить строку				
Власть Мерлот	12345.67	Вино	Алкогольные напитки	Продовольственные продукты
Удалить строку				
КефирУроженский	12345.67	Кефир	Молочные	Продовольственные продукты

Внести

Загрузить
Обзор... Файлы не выбраны.
Очистить

Обработать

Рисунок 8 – Добавление чека

По использованию кнопки «Внести» отображилось уведомление о необходимости заполнить поля с ценами. Осуществлено заполнение недостающими данными. Повторена отправка, после чего запрос был обработан и чек отображился в профиле пользователя.

Таким образом, было проведено полное тестирование ручного добавления чека.

6 ПОДГОТОВКА И ПЕРЕДАЧА ДАННЫХ ДЛЯ СОПРОВОЖДЕНИЯ

Внедрение - это процесс развертывания в рабочем окружении компании заказчика и интеграция с уже существующими информационными системами и бизнес-приложениями.

Акт — это официальный документ, который констатирует произошедшее действие или факт хозяйственной жизни и подписывается уполномоченными должностными лицами. Настоящий акт должен быть составлен в двух экземплярах, один из которых должен находиться у заказчика, а другой у исполнителя. Акт позволяет зафиксировать факт о внедрении программного средства в производство.

Для того чтобы программу внедрить в производство, необходимо заполнить следующий акт.

УТВЕРЖДАЮ

Директор ООО
«Девкрафт»

Яговдик А. С.

«__» _____ 2019 г.

АКТ

о внедрении программного комплекса «CheckbookAccounting» в производство

Настоящий акт составлен об использовании в производстве сайта-сервиса для учета чеков по ряду критериев

(наименование разработки, объекта внедрения)

Разработка использована в производственном процессе «Девкрафт»
Отдел разработки - 25.04.2019 09:00

Описание объекта внедрения прилагается и является неотъемлемой частью Акта.

Директор Яговдик А.С. _____ Руководитель проекта _____

ОПИСАНИЕ ОБЪЕКТА ВНЕДРЕНИЯ

Программа для учета личной бухгалтерии на основе распознавания
визуальных образов чека

(название разработки)

1. Краткая характеристика объекта внедрения и его назначение

Программа для учета личной бухгалтерии на основе распознавания
визуальных образов чека.

2. Фамилия и инициалы разработчиков, место работы, должность
Олишкевич И.Р., учащийся ЧУО «Гродненский колледж бизнеса и права»,
специальность 2-40 01 01 «Программное обеспечение информационных
технологий»

3. Фамилия и инициалы сотрудников, использующих разработку Яговдик
А.С. - директор «Девкрафт»

4. Начало использования объекта внедрения (месяц, год)

26.04.2019 12:00

Директор Яговдик А.С. _____

Разработчик Олишкевич И.Р. _____

7 ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

7.1 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 (Π_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. $V_0 = 35820$ LOC.

Таблица 1 – Перечень и объем функций программного модуля

N функции	Наименование (содержание)	Объем функции (LOC)
1	2	3
103	Анализ входного языка	660
105	Обработка входного заказа и формирование таблицы	1340

Окончание таблицы 1 – Перечень и объем функций программного модуля

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
305	Обработка файлов	800
309	Формирование файла	1080
701	Математическая статистика и прогнозирование	9320
703	Расчет показателей	460
707	Графический вывод результатов	480
	Итого	35820

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

Нормативная трудовоемкость разработки ПО. На основании принятого к расчету объема (V_0) и категории сложности (Прил. 3) определяется нормативная трудовоемкость ПО, $T_n = 950$ человеко-дней.

Нормативная трудовоемкость (T_n) служит основой для определения общей трудовоемкости (T_o), расчет которой осуществляется различными способами в зависимости от размера проекта.

Общая трудовоемкость небольших проектов рассчитывается по форме

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

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 ($K_{сл}$). 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.

$$K_C = 1 + 0,12 + 0,08 + 0,07 = 1,27$$

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:

$$T_o = 950 * 1,27 * 0,9 * 0,8 = 868,7 \text{ чел./дн.}$$

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 ($T_{уз}$, $T_{уэ}$, $T_{ут}$, $T_{ур}$, $T_{ув}$) 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:

$$\text{трудоемкость стадии ТЗ: } T_{уз} = T_n * K_c * d_3 * K_n = 950 * 0,09 * 1,27 * 0,7 = 76 \quad (9)$$

$$\text{трудоемкость стадии ЭП: } T_{уэ} = T_n * K_c * d_3 * K_n = 950 * 0,07 * 1,27 * 0,7 = 59,12 \quad (10)$$

$$\text{трудоемкость стадии ТП: } T_{ут} = T_n * K_c * d_T * K_n = 950 * 0,07 * 1,27 * 0,7 = 59,12 \quad (11)$$

$$\text{трудоемкость стадии РП: } T_{ур} = T_n * K_c * d_p * K_n * K_T = 950 * 0,61 * 1,27 * 0,8 * 0,7 = 412 \quad (12)$$

$$\text{трудоемкость стадии ВН: } T_{ув} = T_n * K_c * d_B * K_n = 950 * 0,16 * 1,27 * 0,8 = 154,43 \quad (13)$$

Общая трудоемкость определяется как сумма трудоемкостей по стадиям:

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

$$T_y = 76 + 59,12 + 59,12 + 412 + 154,43 = 760,67 \text{ человеко-дней}$$

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

Таблица 2 – 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 трудоемкости 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.	85,5	66,5	66,5	579,5	152	950

Окончание таблицы 2 – Расчет общей трудоемкости разработки ПО и

численности исполнителей с учетом стадий

3. Коэффициент сложности ПО (K_c)	1,2 7	1,27	1,27	1,27	1,27	
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.	76	59,12	59,12	412	154,43	760,67

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

7.2 Расчёт эффективного фонда времени и численности работников

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

$$\Phi_{эф} = D_r - D_{п} - D_{в} - D_o \quad (15)$$

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.

$$T_{пл}=228/12*3=57 \text{ дней}$$

7.3 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 трудоемкость, 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 ($T_{м1}$) на тарифный коэффициент (T_k), соответствующий установленному тарифному разряду:

$$T_m = T_{м1} * T_k \quad (17)$$

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 (Φ_p):

$$T_{ч} = T_m / \Phi_p \quad (18)$$

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

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

Основная заработная плата исполнителей на конкретное ПО рассчитывается по формуле:

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

где n – количество исполнителей, занятых разработкой конкретного ПО;

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

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

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

K – коэффициент премирования.

В соответствии со штатным расписанием на разработке будут заняты:

- 1) программист 2 категории – тарифный разряд – 12; тарифный коэффициент – 2.84;
- 2) программист 1 категории – тарифный разряд – 14; тарифный коэффициент – 3.25;
- 3) ведущий программист – тарифный разряд – 15; тарифный коэффициент – 3.48;
- 4) начальник лаборатории – тарифный разряд – 16; тарифный коэффициент – 3.72;

Расчётные данные представим в виде таблицы.

Таблица 3 – 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:

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

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.

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

$H_{бгс}$ – 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.

7.4 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.

7.5 Расчёт расходов на оплату машинного времени

Расходы по статье «Машинное время» (P_{mi}) включают оплату машинного времени, необходимого 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).

$$P_{mi} = 0,2 \cdot 35820 / 100 \cdot 12 = 850,68 \text{ рублей}$$

7.6 Расчет прочих затрат

Расходы по статье «Прочие затраты» (Π_{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:

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

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

Π_z – прочие затраты в целом по организации.

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

7.7 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 4 – 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	$Z_{сzi}$	234
Отчисления в Белгосстрах	$Z_{бгс}$	4,1
Материалы	M_i	17,2
Машинное время	P_{mi}	85,9

Окончание таблицы 4 – Смета затрат

Прочие затраты	P_{zi}	114,7
Итого производственная себестоимость	$C_{пр}$	1144,2

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

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

$C_{пр}$ – производственная себестоимость;

$H_{са}$ – норматив отчисления на сопровождение и адаптацию программного средства (Прил. 7).

$$P_{ci} = 1144,2 * 20 / 100 = 228,8 \text{ рублей}$$

Таким образом полная себестоимость разработки программного средства определяется по формуле:

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

$$C_{пол} = 1144,2 + 228,8 = 1373 \text{ рублей}$$

7.8 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:

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

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

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

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

$$\Pi = 1373 * 20 / 100 = 274,6 \text{ рублей}$$

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

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

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

Π – прибыль

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

$$\text{НДС} = (1373 + 274,6) * 20 / 100 = 329,5 \text{ рублей}$$

Результат расчётов заносится в таблицу.

Таблица 5 – Прогнозируемая отпускная цена

Наименование статей затрат	Обозначение	Сумма
Полная производственная себестоимость	Спол	1373
Прибыль и рентабельность по создаваемому программному средству	П	274,6
Налог на добавленную	НДС	329,5
Прогнозируемая отпускная цена	Ц	1977,1

7.9 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:

$$П_{рп} = B_p - C - НДС \quad (35)$$

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

Налог на добавленную стоимость рассчитывается по формуле:

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

где B_p – выручка от реализации продукции;

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

$$НДС = 1977,1 * 20 / (100 + 20) = 329,5 \text{ рублей}$$

$$П_{рп} = 1977,1 - 1373 - 329,5 = 274,6 \text{ рублей}$$

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).

$$\text{Налог на прибыль} = 274,6 * 18 / 100 = 49,4 \text{ рублей}$$

$$ЧП = 274,6 - 49,4 = 225,2 \text{ рублей}$$

7.10 Расчёт показателей эффективности от внедрения программного средства

К показателям эффективности от внедрения программного средства относятся:

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

Срок окупаемости рассчитывается по формуле:

$$T_{ок} = Z / ЧП \quad (38)$$

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

ЧП – чистая прибыль.

$$T_{ок} = 1373 / 225,2 = 6 \text{ месяца}$$

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

$$K_{эф} = 1 / T_{ок} \quad (39)$$

где $T_{ок}$ – срок окупаемости проекта

$$K_{эф} = 1 / 6 = 0.16$$

Рентабельность затрат рассчитывается по формуле:

$$P_z = ЧП / Z * 100 \quad (40)$$

где ЧП – чистая прибыль;

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

$$P_z = 225,2 / 1373 * 100 = 16\%$$

ЗАКЛЮЧЕНИЕ

Задачей преддипломной практики являлась разработка web-сервера для организации rest full api, взаимодействующего с android-клиентом, представляя комплекс программного обеспечения по восприятию визуальных образов чека с дальнейшим преобразованием в составлении учета личной бухгалтерии потребителя.

В ходе выполнения задания практики, были разработаны программный средства с использованием языков программирования «Java», «Python» и «JavaScript», а также с помощью языка структурированных запросов «SQL». Усовершенствованы навыки в данных языках. Это позволило освоить и закрепить, расширить и систематизировать знания, полученные во время изучения специальных дисциплин.

Была реализована база данных на основе подхода «CodeFirst» посредством «SQL-Alchemy» и 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.

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.

Разработанное обеспечение быстро и безошибочно справляется с поставленной задачей хранения и обработки информации. В удобном интерфейсе воплощены все необходимые для данной работы возможности.

Основные модификации по дальнейшему улучшению заключаются в:

- 1) реинжиниринге;

- 2) интернационализации и локализации;
- 3) портировании и миграции программного обеспечения.

Достоинством проекта является простота и интуитивность программного средства.

Исходя из вышесказанного можно утверждать о выполненной цели преддипломной практики. Данный проект реализовал поставленные задачи в соответствии с заданным условием.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

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. – 416 с.
10. Любанович, Б. Простой Python. Современный стиль программирования / Б. Любанович. – Питер, 2017. – 480с.
11. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. – Издательство «Символ», 2015. – 608с.
12. Лутц, М. Python. Карманный справочник /М. Лутц. – Издательство «Вильямс», 2015. – 320с.

13. Гринберг, М. Разработка веб-приложений с использованием Flask на языке Python / М. Гринберг – Издательство «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.

ПРИЛОЖЕНИЕ А

(справочное)

Экранные формы

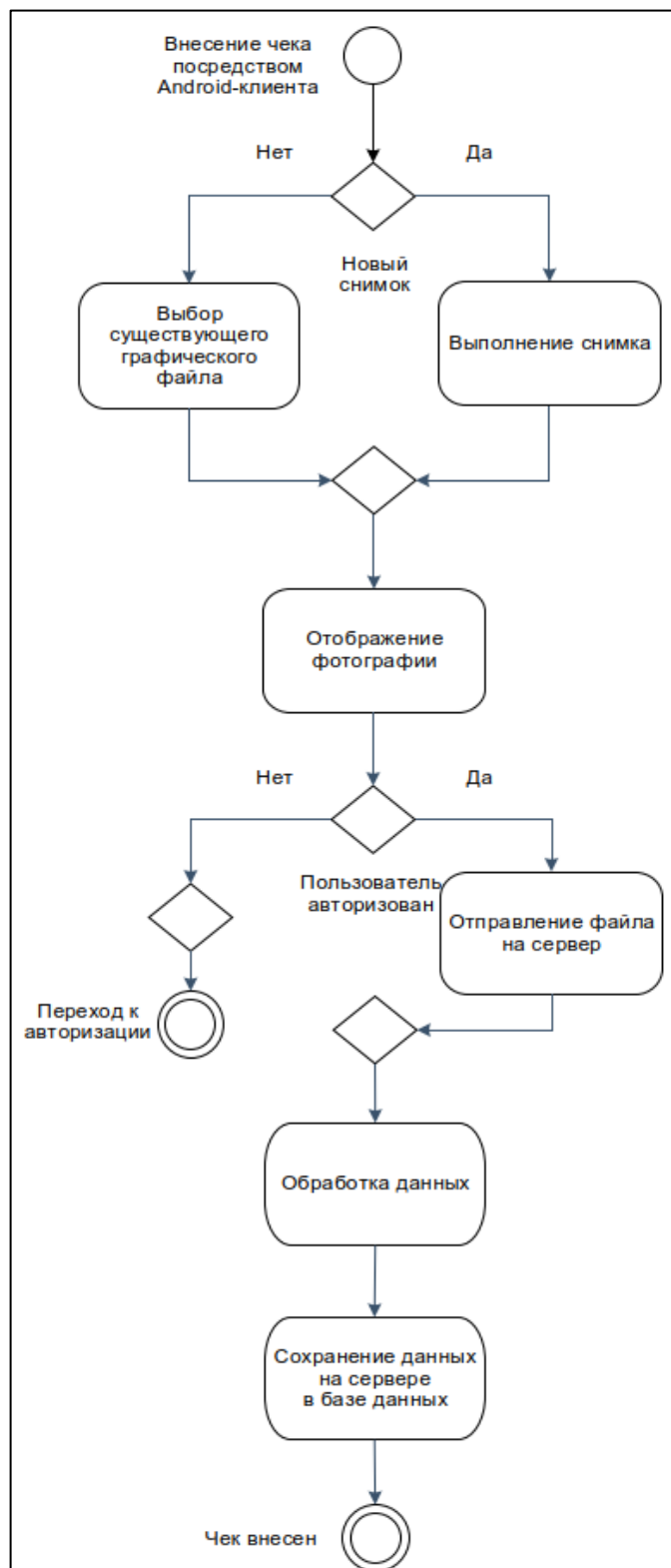


Рисунок А.1– Диаграмма деятельности

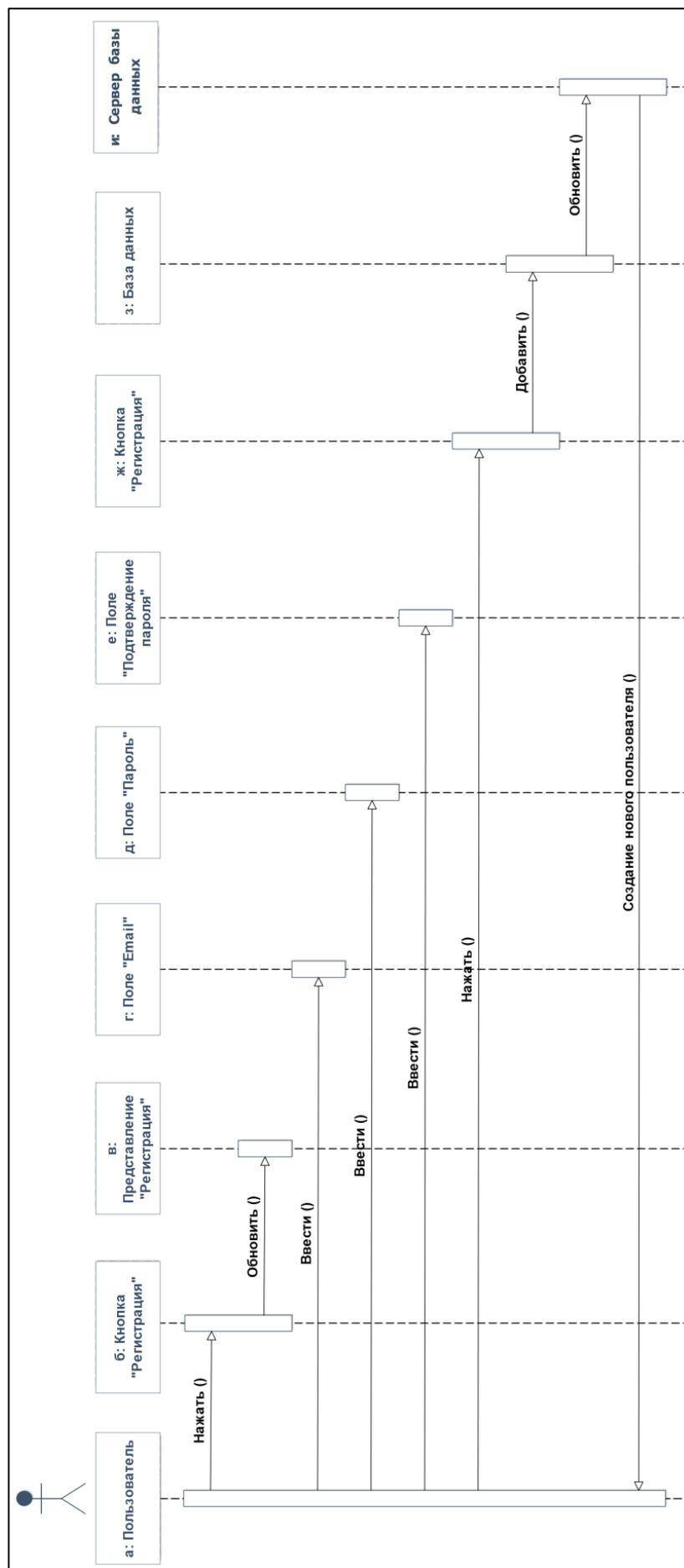


Рисунок А.2 – Диаграмма последовательности действий

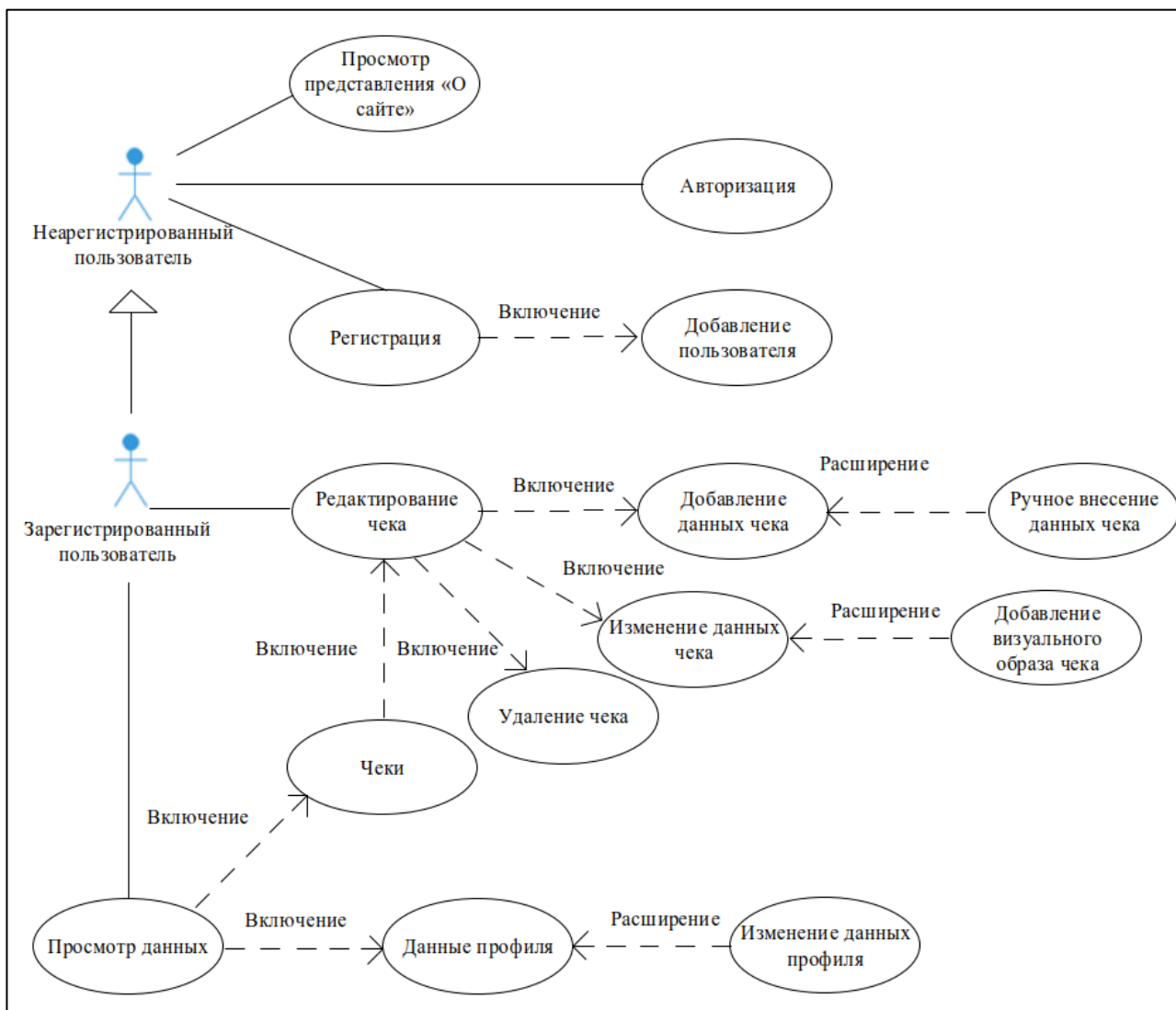


Рисунок А.3 – Диаграмма вариантов использования

ПРИЛОЖЕНИЕ Б

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
from project.controllers.api.v1 import user_controller, organization_controller, product_controller
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_abbey
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)
            startabbyy(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("УНП") != -1:
unp_str = upper_text[upper_text.rfind("УНП") + 3:upper_text.rfind("УНП") + 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])
            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("ПП.") != -1 or upper_text.find("УЛ.") != -1 or upper_text.find("Г.") != -1 \
        or upper_text.find("ПП-Т") != -1 or upper_text.find("ПП-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('СКО') != -1 or line.find('ПЕГ') != -1 or line.find('КАССА') != -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('ДОК') != -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

```