Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

ПО РАЗРАБОТКЕ И СОПРОВОЖДЕНИЮ

ОП Т. 895017

Руководитель практики ( Е.В.Багласова )

Учащийся ( А.А. Петровский )

2021

Изм.

Лист

№ докум.

Подпись

Дата

Лист

У

ОП Т.895017

Разраб.

*А.А. Петровский*

Провер.

Е.В. Багласова

Т. Контр.

Н. Контр.

Утверд.

Отчет по учебной практике по разработке и сопровождению

Лит.

Листов

33

КБП

Содержание

[1 Программа практики 3](#_Toc76505794)

[2 Объектно-ориентированный анализ и проектирование приложения 4](#_Toc76505809)

[2.1 Сущность задачи 4](#_Toc76505810)

[2.2 Проектирование модели 4](#_Toc76505811)

[3 Вычислительная система 7](#_Toc76505812)

[3.1 Требования к аппаратным и операционным ресурсам 7](#_Toc76505813)

[3.2 Инструменты разработки 7](#_Toc76505814)

[4. Проектирование приложения 8](#_Toc76505815)

[4.1 Требования к приложению и её наполнению 8](#_Toc76505816)

[4.2 Структура приложения 8](#_Toc76505817)

[4.3 Концептуальный прототип 8](#_Toc76505818)

[4.4 Организация данных 11](#_Toc76505819)

[5 Реализация приложения 12](#_Toc76505820)

[5.1 Реализация поведения объектов 12](#_Toc76505821)

[6 Описание приложения 15](#_Toc76505822)

[6.1 Технические требования 15](#_Toc76505823)

[6.2 Функциональное назначение 15](#_Toc76505824)

[7 Методика испытаний 16](#_Toc76505825)

[7.1 Технические требования 16](#_Toc76505826)

[7.2 Функциональное тестирование 16](#_Toc76505827)

[8 Применение 20](#_Toc76505828)

[Заключение 21](#_Toc76505829)

[Список информационных источников 22](#_Toc76505830)

[Приложение А 23](#_Toc76505831)

# 1 Программа практики

# Целями практики по разработке и сопровождению программного обеспечения являются:

# - закрепление знаний, связанных с технологией обработки информации;

# - приобретение навыков разработки программ с использованием современных средств обработки экономической и деловой информации;

# - выработка умений применять средства стандартных библиотек в практических задачах.

# Задачами практики по разработке и сопровождению ПО являются:

# - углубленное изучение принципов организации программного обеспечения и

# технологии его проектирования;

# - владение методами надежного программирования;

# - умение разрабатывать программы в соответствии с промышленными требованиями,

# обеспечивая высокий уровень качества программного обеспечения и экономической

# эффективности;

# - изучение способов определения экономической себестоимости и эффективности

# разработки программного обеспечения;

# - оформление комплекта документации на созданное программное обеспечение.

# 2 Объектно-ориентированный анализ и проектирование приложения

## 2.1 Сущность задачи

Приложение «Input» предназначено для автоматизации принятия и создания заявок на создание приложения. Приложение позволит уменьшить временные затраты на создание и обработку заявки. Обеспечивает автоматическую печать документа в Word.

Из всех задач, которые будет решать разрабатываемое программное средство, можно выделить ряд основных:

- ведение базы данных, содержащей информацию о клиентах, программистах;

- осуществление контроля созданных заявок;

- создание документа о заказе в Word;

- просмотр требований клиента к приложению;

- принятие заявки на создание приложение;

- создание заявки на создание приложения.

Данное программное средство направлено на упрощение работы программиста. Такого рода программы упрощают работу, связанную с принятием заказа, просмотр требований. Разрабатываемая программа будет автоматически рассчитывать создавать таблицу с созданными заказами, их статусом и исполнителем.

Аналога данного программного средства представлено не было.

## 2.2 Проектирование модели

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD), которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы. Основными понятиями данной нотации являются понятия сущности и связи. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Связь определяется как отношение или некоторая ассоциация между отдельными сущностями.

Графическая модель данных строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей. Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Для проекта можно выделить следующие сущности: «Клиент», «Программист», «Заказ».

Для сущности «Клиент» атрибутами будут являться:

– логин;

– ФИО;

­– пароль.

Для сущности «Программист» атрибутами будут являться:

– логин;

– ФИО;

– пароль;

Для сущности «Заказ» атрибутами будут являться:

– идентификационный код;

– клиент;

­– статус;

– заголовок;

– описание;

­– бюджет;

– программист.

Информационная модель базы представлена на диаграмме «Сущность-связь» на рисунке 2.1.

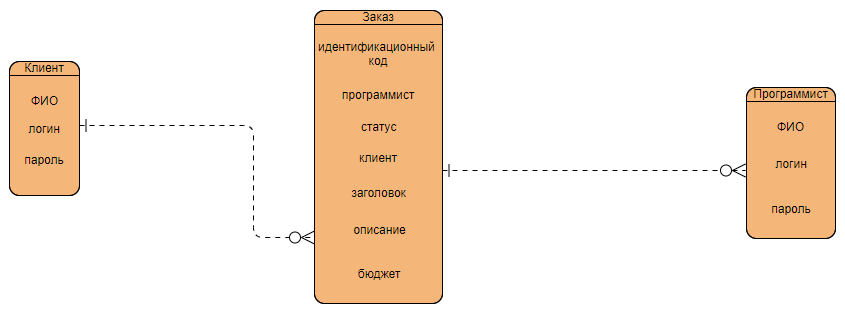


Рисунок 2.1 – Диаграмма «Сущность-связь»

В рамках унифицированного языка моделирования (UML) все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграмма деятельности, диаграмма последовательности, диаграмма компонентов.

Перечень этих диаграмм представляет собой неотъемлемую часть графической нотации языка UML, сам процесс объектно-ориентированного программирования (ООП) неразрывно связан с процессом построения этих диаграмм. Совокупность построенных таким образом диаграмм содержит всю информацию, необходимую для реализации проекта сложной системы.

Диаграмма – граф специального вида, состоящий из вершин в форме геометрических фигур, которые связаны между собой ребрами или дугами.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы.

В данной проектируемой системе в качестве актера выступает пользователь, который служит источником воздействия на моделируемую систему.

К основным функциям разрабатываемой программы относятся следующие функции:

* регистрация;
* авторизация;
* отображение списка заказов;
* создание нового заказа;
* отображение информации о заказе;
* принятие заказа программистом;
* печать документа.

Диаграмма вариантов использования представлена на листе 1 графической части.

Центральное место в ООП занимает разработка логической модели системы в виде диаграммы классов. Нотация классов в языке UML проста и интуитивно понятна. Схожая нотация применяется и для объектов — экземпляров класса, с тем различием, что к имени класса добавляется имя объекта, и вся надпись подчеркивается.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами кото­рого являются элементы типа «классификатор» и которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят от времени.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому.

Обязательным элементов обозначения класса является его имя. На начальных этапах разработки диаграммы отдельный класс может обозначаться простым прямоугольником с указанием только имени соответствующего класса. По мере проработки отдельных компонентов диаграммы классов дополняются атрибутами и операциями. Предполагается, что окончательный вариант диаграммы содержит наиболее полное описание классов, которые состоят из трех разделов или секций. Диаграмма классов представлена на листе 4 графической части.

Диаграммы деятельности – частный случай диаграмм состояний. Основная цель использования таких диаграмм – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности для данной проектируемой системы представлена на листе 1 графической части.

Для моделирования взаимодействия объектов в UML используется диаграмма последовательности. Диаграмма последовательности для данной проектируемой системы представлена на листе 5 графической части.

Все предыдущие диаграммы отражали концептуальные аспекты построения модели системы и относились к логическому уровню представления и оперировали понятиями, которые не имеют самостоятельного материального воплощения. Для описания реальных сущностей предназначен другой аспект модельного представления, а именно физическое представление модели. Для физического представления модели используются диаграмма компонентов.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Диаграмма компонентов для данной проектируемой системы представлена на листе 5 графической части.

# 3 Вычислительная система

## 3.1 Требования к аппаратным и операционным ресурсам

Конфигурация персонального компьютера, на котором будет разрабатываться программа:

* процессор AMD Ryzen 5 2600;
* оперативная память DDR4-3600 МГц, 16 ГБ;
* видеокарта GeForce GTX 1050TI;
* твердотельный накопитель SDD или HDD;
* клавиатура и мышь.

## 3.2 Инструменты разработки

Программное обеспечение, используемое для разработки программного средства:

– операционная система Windows 10;

– среда разработки Visual Studio 2019;

– язык программирования С#;

– технология WPF;

– программа для разработки справочной информации Microsoft Word 2018;

– сase-средство для построения диаграмм StarUML;

­– средство разработки баз данных Microsoft SQL Server Management Studio 2018.

Windows 10 – [операционная система](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) семейства [Windows NT](https://ru.wikipedia.org/wiki/Windows_NT) производства корпорации [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Предназначена для [рабочих станций](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B1%D0%BE%D1%87%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D0%BD%D1%86%D0%B8%D1%8F), [персональных компьютеров](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) и портативных устройств; версия, предназначенная для решения серверных задач.

Microsoft Visual Studio 2019 – это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

C# – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. В данном программном средстве не предусмотрена возможность сетевой поддержки.

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Программа Microsoft Word 2016 – текстовый процессор, предназначенный для создания, просмотра и редактирования текстовых документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office.

# 4. Проектирование приложения

## 4.1 Требования к приложению и её наполнению

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Для того, чтобы программа выглядела более привлекательно нужно воспользоваться сторонним графическим пользовательским интерфейсом Material Design Theme.

Кроме того, при разработке форм необходимо соблюдать определённые требования: формы взаимодействия с клиентом не должны содержать лишней информации. При конструировании форм в необходимых случаях нужно предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных. Приложение должно иметь страничную структуру, то есть все изменения интерфейса должны происходить с использованием одного окна. Характеристики ПК, на котором производится тестирование:

* версия операционной системы Windows 10;
* разрешение экрана 1920x1080;
* процессор Intel Core-i3-9100U 3.6 ГГц;
* оперативная память DDR3-1600 МГц, 8 ГБ;
* AMD Radeon Vega Graphics;
* твердотельный накопитель SDD или HDD;

## 4.2 Структура приложения

В данной приложении клиент может оставить заказ на создание приложения. Клиенту предоставляется возможность оформления заявки, просмотр оформленных и их статуса. Программисту предоставляется возможность принять заказ и распечатать заказ.

В данной игре присутствуют следующие окна:

* Авторизация;
* Регистрация;
* Окно программиста;
* Окно клиента.

Окно «Регистрация» представляет собой форму для регистрации клиента или программиста.

Окно «Авторизация» представляет собой форму для авторизации клиента или программиста.

Окно программиста представляет собой форму для просмотра информации о заказе и принятии данного заказа.

Окно клиента представляет собой форму для создания заявки и просмотра существующих

## 4.3 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – систему кнопок и форм.

Главное окно программы будет представлено формой Login, на которой находятся основные элементы управления программой. С помощью полей на главной форме можно ввести данные для авторизации и нажать кнопку «Вход». Главное окно программы, в котором располагаются основные кнопки, представлено на рисунке 4.1.

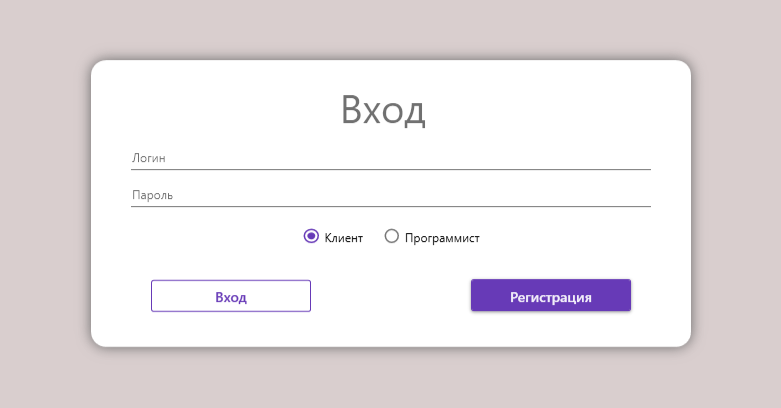


Рисунок 4.1 – Главное окно программы

Регистрация происходит путем заполнения форм и нажатия кнопки «Регистрация». Данное окно представлено на рисунке 4.2.

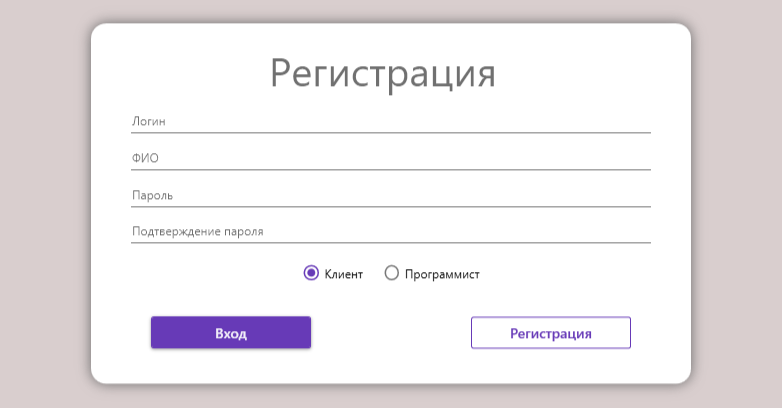


Рисунок 4.2 – Окно регистрации

Окно клиента предоставляет возможность составить заявку и посмотреть созданные. Данное окно представлено на рисунке 4.3.

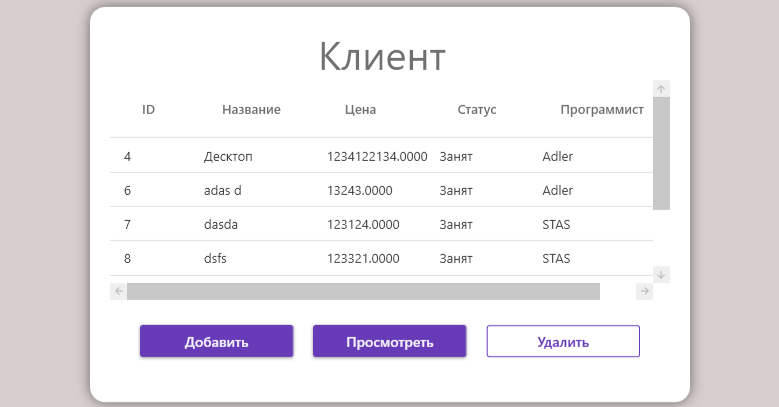


Рисунок 4.3 – Окно клиента

Окно программиста предоставляет просмотреть заявку и принять ее. Данное окно представлено на рисунке 4.4.

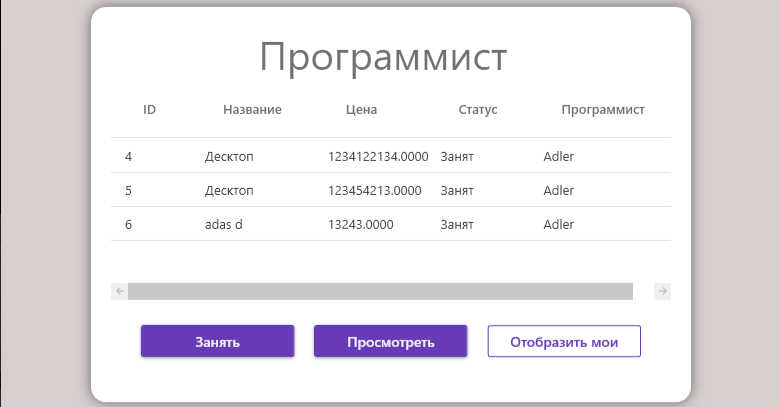


Рисунок 4.4 – Окно программиста

## 4.4 Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства включает три таблицы: Заказ, Клиент, Программист.

# 5 Реализация приложения

## 5.1 Реализация поведения объектов

В данном приложении необходимо обеспечить работу следующего функционала:

* Создание заявки;
* Принятие заявки;
* Авторизация;
* Регистрации.

В следующем участке кода описан весь функционал окна Авторизации.

public partial class Login : Page

{

Criterion crt;

public Login()

{

InitializeComponent();

}

private enum Criterion

{

customer,

programmer

}

private void RadioButton\_Checked(object sender, RoutedEventArgs e)

{

RadioButton pressed = (RadioButton)sender;

string s = pressed.Content.ToString();

if (s == "Клиент")

{

crt = Criterion.customer;

return;

}

if (s == "Программист")

{

crt = Criterion.programmer;

return;

}

}

private void Button\_Login\_Click(object sender, RoutedEventArgs e)

{

string login = loginBox.Text.Trim().ToUpper();

string password1 = firstPassBox.Password.Trim();

bool isGood = true;

if (login.Length < 1)

{

isGood = false;

loginBox.ToolTip = "Введите логин!";

loginBox.Foreground = Brushes.Red;

}

else

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

}

if (password1.Length < 1)

{

isGood = false;

firstPassBox.ToolTip = "Введите пароль!";

firstPassBox.Foreground = Brushes.Red;

}

else

{

firstPassBox.ToolTip = " ";

firstPassBox.Foreground = Brushes.Black;

}

if (isGood == true)

{

if(crt == Criterion.customer)

{

DataTable table = SQLbase.Select($"select \* from Customer where login = '{login}'");

if (table.Rows.Count > 0)

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

table = SQLbase.Select($"select \* from Customer where login = '{login}' and pass = '{password1}'");

if (table.Rows.Count > 0)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

CustomerPage nextPage = new CustomerPage(login);

nav.Navigate(nextPage);

}

else

{

firstPassBox.ToolTip = "Неправильный пароль!";

firstPassBox.Foreground = Brushes.Red;

}

}

else

{

loginBox.ToolTip = "Такого пользователя не существует!";

loginBox.Foreground = Brushes.Red;

}

}

if (crt == Criterion.programmer)

{

DataTable table = SQLbase.Select($"select \* from Programmer where login = '{login}'");

if (table.Rows.Count > 0)

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

table = SQLbase.Select($"select \* from Programmer where login = '{login}' and pass = '{password1}'");

if (table.Rows.Count > 0)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

ProgrammerPAge nextPage = new ProgrammerPAge(login);

nav.Navigate(nextPage);

}

else

{

firstPassBox.ToolTip = "Неправильный пароль!";

firstPassBox.Foreground = Brushes.Red;

}

}

else

{

loginBox.ToolTip = "Такого пользователя не существует!";

loginBox.Foreground = Brushes.Red;

}

}

}

}

private void GoToReg(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("/Register.xaml", UriKind.Relative));

}

}

# 6 Описание приложения

## 6.1 Технические требования

Минимальными требованиями для оптимальной работы программного средства является

персональный компьютер со следующими характеристиками:

– процессор 1500 МГц и выше;

– оперативная память 800 Мбайт и более;

– свободное место на диске 700 Мбайт;

– монитор;

– мышь, клавиатура;

– принтер.

Компьютер должен работать под управлением операционной системы, начиная с

Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний

является Windows 7, так как она ориентирована на максимальное использование всех

возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

## 6.2 Функциональное назначение

Приложение «Input» предназначено для автоматизации принятия и создания заявок на создание приложения. Приложение позволит уменьшить временные затраты на создание и обработку заявки. Обеспечивает автоматическую печать документа в Word.

База данных проекта содержит необходимые таблицы для реализации поставленных задач. Предоставлена возможность добавления новых программистов и клиентов. Создание заявки, ее подтверждение, просмотр и печать документа.

# 7 Методика испытаний

## 7.1 Технические требования

Данную игру необходимо реализовать с использованием лаконичного и с использованием стилей интерфейса. Дизайн приложения должен быть простым и понятным пользователю.

Минимальные системные требования к приложению: ПК, экран и клавиатура.

Характеристики ПК, на котором производится тестирование:

* версия операционной системы Windows 10;
* разрешение экрана 1920x1080;
* процессор Intel Core-i3-9100U 3.6 ГГц;
* оперативная память DDR3-1600 МГц, 8 ГБ;
* AMD Radeon Vega Graphics;
* твердотельный накопитель SDD или HDD;
* клавиатура и мышь.

## 7.2 Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствие с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывались на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация по пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

В процессе разработки игры базовые механики подвергаются проверке. Требуется убедится в том, что все нажатия обрабатываются верно и безошибочно.

В таблицах 7.1 – 7.4 представлены тест-кейсы, подготовленные для проведения функционального тестирования.

Таблица 7.1 – Тест-кейс проверки окна регистрации

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 001 | Регистрация | 1. Запустить приложение.  2. Нажать на кнопку регистрации.  3. Заполнить поля.  4. Нажать на кнопку регистрация. | Ожидаемый результат: Отобразится окно авторизации. |
| Фактический результат: Отображено окно авторизации.  Результат представлен на рисунке 7.1. |

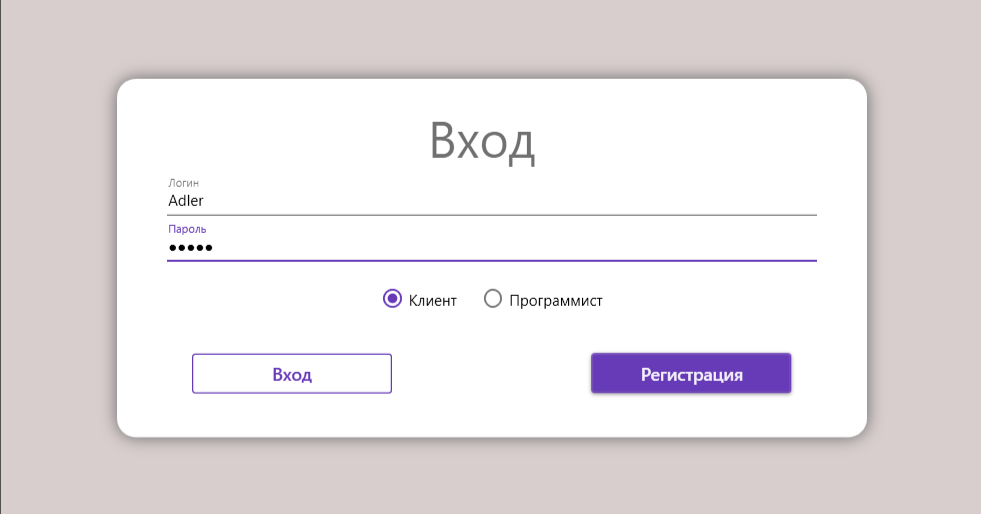


Рисунок 7.1 - Окно авторизации

Таблица 7.2 – Тест-кейс проверки окна авторизации

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 002 | Авторизация | 1. Запустить приложение.  2. Заполнить поля.  4. Нажать на кнопку вход. | Ожидаемый результат:  Отобразится окно клиента или программиста в зависимости от того, кто проходил авторизацию.  Фактический результат: отображено окно клиента или программиста в зависимости от того, кто проходил авторизацию.  Результат представлен на рисунке 7.2. |

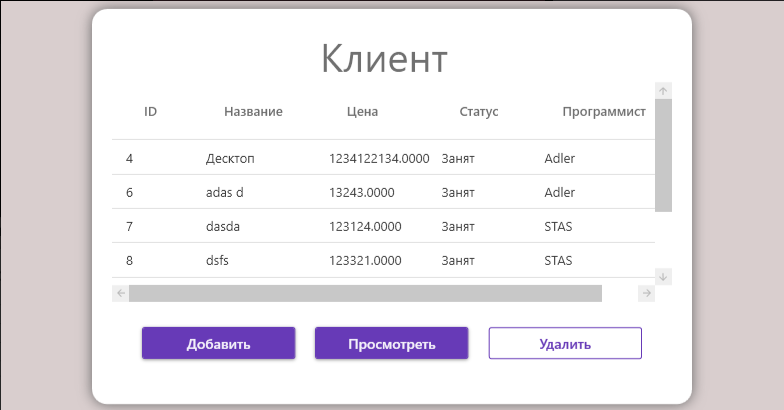


Рисунок 7.2 – Окно клиента

Таблица 7.3 – Тест-кейс проверки функции «Добавление заказа»

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 003 | Заказ | 1. Запустить приложение  2. Пройти авторизацию под клиентом  3. Нажать на кнопку добавления заказа | Ожидаемый результат: открывается форма добавления заказа |
| Фактический результат: открыта форма добавления заказа Результат представлен на рисунке 7.3. |

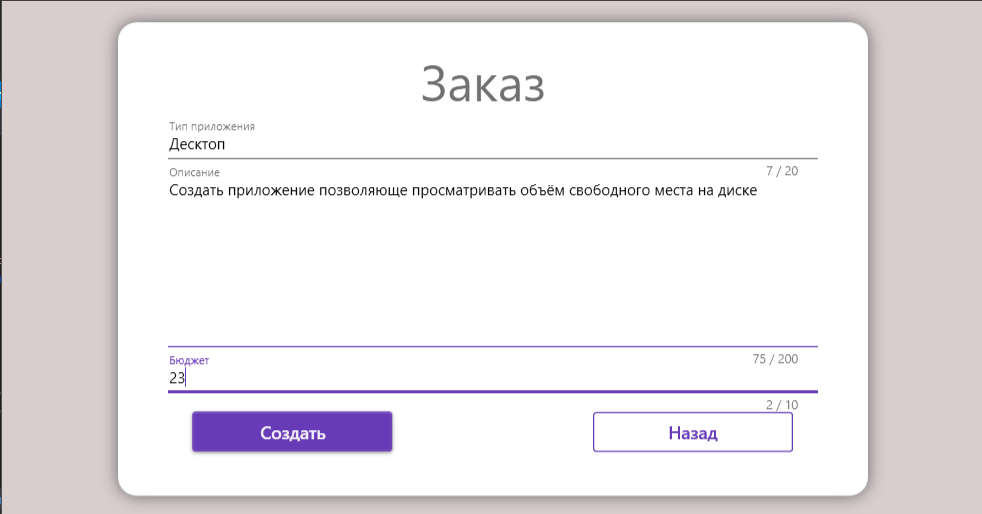


Рисунок 7.3 – Добавление заказа

Таблица 7.4 – Тест-кейс проверки функции «Просмотр заказа»

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Модуль/Функция | Шаги воспроизведения | Результат |
| 004 | Просмотр заказа | 1. Запустить приложение  2. Пройти авторизацию под программистом  3. Выбрать заказ  4. Нажать на кнопку просмотреть | Ожидаемый результат: открывается информация о заказе |
| Фактический результат: открыта информация о заказе Результат представлен на рисунке 7.4. |

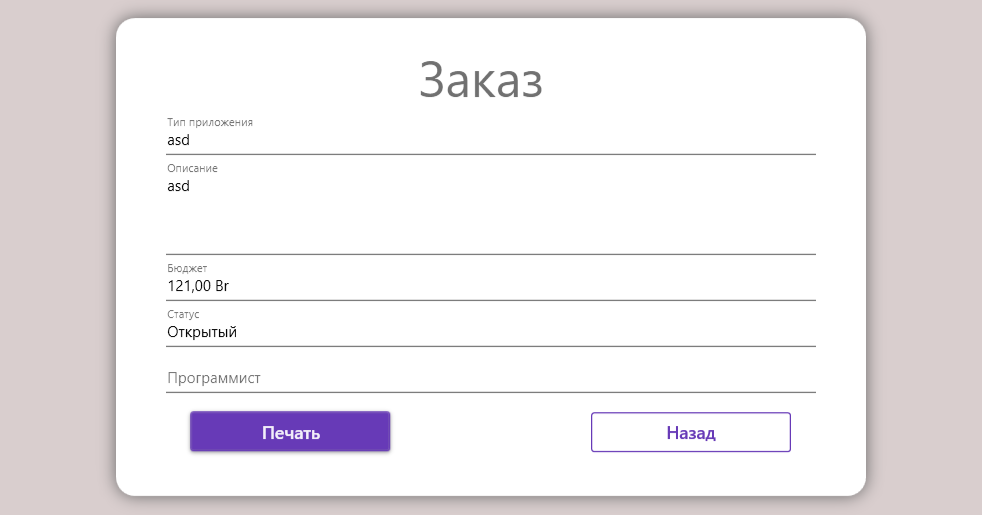


Рисунок 7.4 – Информация о заказе

# Применение

Программа предназначена для Fullstack программистов и клиентов. Это приложение позволяет комфортно просматривать и принимать заявки. Клиенту предоставляется возможность создавать и удалять заявки.

Для применения данного программного средства необходимы следующие технические требования:

– процессор 1500 МГц и выше;

– оперативная память 800 Мбайт и более;

– свободное место на диске 700 Мбайт;

– монитор;

– мышь, клавиатура;

– принтер.

# Заключение

Программа предназначена для создания заявки клиентом и обработки заявки программистом. Это приложение позволяет комфортно создавать, просматривать и принимать заявки. Для достижения цели курсового проектирования были решены следующие задачи:

– определена вычислительная система, необходимая для создания программного средства;

– разработаны логическая и физическая модели данных; – по модели выполнено проектирование задачи;

– разработано программное средство;

– описано созданное программное средство;

– выбрана методика испытания;

– описан процесс функционального тестирования;

– приведены примеры области применения.

Разработка имеет интуитивно понятный интерфейс, позволяющий с минимальным знанием компьютера использовать данное программное средство.

В процессе разработки данного программного средства были применены и закреплены знания по уже изученному материалу, были отработаны навыки владения методами надёжного программирования и эффективности разработки программного обеспечения в «Visual Studio» с использованием языка программирования C#.

Программа реализована в полном объёме, в соответствии с техническим заданием, полностью отлажена и протестирована. Поставленные задачи выполнены.

# Список информационных источников

1. Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2021.

2 Draw.io – бесплатное средство для создания блок-схем, инфографики, прототипов [Электронный ресурс]. – EL-BLOG.RU, 2021 – Режим доступа : <https://el-blog.ru/draw-io/>. – Дата доступа : 20.03.2021.

3   Общие сведения о Visual Studio [Электронный ресурс]. – Microsoft, 2021 – Режим доступа : <https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019>. – Дата доступа : 10.03.2021.

4   Что такое ER-диаграмма и как ее создать? [Электронный ресурс]. – Lucid Software Inc, 2021 – Режим доступа : <https://www.lucidchart.com/pages/ru/erd-диаграмма>. – Дата доступа : 15.03.2021.

5  Диаграммы классов UML. Логическое моделирование [Электронный ресурс]. – INFORMICUS, 2021 – Режим доступа: <http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=3>. – Дата доступа: 15.03.2021.

6   Диаграмма вариантов использования (use case diagram) [Электронный ресурс]. – Леоненков. Самоучитель UML, 2021 – Режим доступа : <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html>. – Дата доступа: 20.03.2021.

7  Диаграмма деятельности (activity diagram) [Электронный ресурс]. – Леоненков. Самоучитель UML, 2020 Режим доступа: <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl7/gl7.html>. – Дата доступа 15.03.2021.

8 Диаграммы компонентов [Электронный ресурс]. – Учебная и научная деятельность Анисимова Владимира Викторовича, 2020. – Режим доступа : <https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema15/tema15_2>. – Дата доступа : 18.03.2021.

9 Краткий обзор языка C# [Электронный ресурс]. – Microsoft, 2021. – Режим доступа : <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>. – Дата доступа : 15.03.2021.

# Приложение А

(Обязательное)

Текст программных модулей

public partial class Login : Page

{

Criterion crt;

public Login()

{

InitializeComponent();

}

private enum Criterion

{

customer,

programmer

}

private void RadioButton\_Checked(object sender, RoutedEventArgs e)

{

RadioButton pressed = (RadioButton)sender;

string s = pressed.Content.ToString();

if (s == "Клиент")

{

crt = Criterion.customer;

return;

}

if (s == "Программист")

{

crt = Criterion.programmer;

return;

}

}

private void Button\_Login\_Click(object sender, RoutedEventArgs e)

{

string login = loginBox.Text.Trim().ToUpper();

string password1 = firstPassBox.Password.Trim();

bool isGood = true;

if (login.Length < 1)

{

isGood = false;

loginBox.ToolTip = "Введите логин!";

loginBox.Foreground = Brushes.Red;

}

else

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

}

if (password1.Length < 1)

{

isGood = false;

firstPassBox.ToolTip = "Введите пароль!";

firstPassBox.Foreground = Brushes.Red;

}

else

{

firstPassBox.ToolTip = " ";

firstPassBox.Foreground = Brushes.Black;

}

if (isGood == true)

{

if(crt == Criterion.customer)

{

DataTable table = SQLbase.Select($"select \* from Customer where login = '{login}'");

if (table.Rows.Count > 0)

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

table = SQLbase.Select($"select \* from Customer where login = '{login}' and pass = '{password1}'");

if (table.Rows.Count > 0)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

CustomerPage nextPage = new CustomerPage(login);

nav.Navigate(nextPage);

}

else

{

firstPassBox.ToolTip = "Неправильный пароль!";

firstPassBox.Foreground = Brushes.Red;

}

}

else

{

loginBox.ToolTip = "Такого пользователя не существует!";

loginBox.Foreground = Brushes.Red;

}

}

if (crt == Criterion.programmer)

{

DataTable table = SQLbase.Select($"select \* from Programmer where login = '{login}'");

if (table.Rows.Count > 0)

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

table = SQLbase.Select($"select \* from Programmer where login = '{login}' and pass = '{password1}'");

if (table.Rows.Count > 0)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

ProgrammerPAge nextPage = new ProgrammerPAge(login);

nav.Navigate(nextPage);

}

else

{

firstPassBox.ToolTip = "Неправильный пароль!";

firstPassBox.Foreground = Brushes.Red;

}

}

else

{

loginBox.ToolTip = "Такого пользователя не существует!";

loginBox.Foreground = Brushes.Red;

}

}

}

}

private void GoToReg(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("/Register.xaml", UriKind.Relative));

}

}

public partial class Register : Page

{

Criterion crt;

public Register()

{

InitializeComponent();

}

private enum Criterion

{

customer,

programmer

}

private void RadioButton\_Checked(object sender, RoutedEventArgs e)

{

RadioButton pressed = (RadioButton)sender;

string s = pressed.Content.ToString();

if (s == "Клиент")

{

crt = Criterion.customer;

return;

}

if (s == "Программист")

{

crt = Criterion.programmer;

return;

}

}

private void Button\_Reg\_Click(object sender, RoutedEventArgs e)

{

string login = loginBox.Text.Trim().ToUpper();

string[] name = nameBox.Text.Trim().Split(' ');

string password1 = firstPassBox.Password.Trim();

string password2 = secondPassBox.Password.Trim();

bool isGood = true;

//Проверка поля логина

if (login.Length < 3 && login.Length > 20)

{

isGood = false;

loginBox.ToolTip = "Длина логина должна быть от 3 до 20 букв!";

loginBox.Foreground = Brushes.Red;

}

else

{

bool stop = false;

foreach (char x in login)

{

if (Char.IsDigit(x))

{

stop = true;

break;

}

if (Convert.ToInt16(x) < 0 || Convert.ToInt16(x) > 128)

{

stop = true;

break;

}

}

if (stop)

{

isGood = false;

loginBox.ToolTip = "Недопустимый ввод символов!";

loginBox.Foreground = Brushes.Red;

}

else

{

if (crt == Criterion.customer)

{

DataTable table = SQLbase.Select($"select \* from Customer where login = '{login}'");

if (table.Rows.Count > 0)

{

isGood = false;

loginBox.ToolTip = "Такой пользователь уже существует!";

loginBox.Foreground = Brushes.Red;

}

else

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

}

}

if(crt == Criterion.programmer)

{

DataTable table = SQLbase.Select($"select \* from Programmer where login = '{login}'");

if (table.Rows.Count > 0)

{

isGood = false;

loginBox.ToolTip = "Такой пользователь уже существует!";

loginBox.Foreground = Brushes.Red;

}

else

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

}

}

}

}

//Поле фио

if (!(name.Length == 2 || name.Length == 3))

{

isGood = false;

nameBox.ToolTip = "Требуется полное написание ФИО!";

nameBox.Foreground = Brushes.Red;

}

else

{

bool stop = false;

foreach (char x in nameBox.Text)

{

if (Char.IsDigit(x))

{

stop = true;

break;

}

}

if (stop)

{

isGood = false;

nameBox.ToolTip = "Недопустимый ввод символов!";

nameBox.Foreground = Brushes.Red;

}

else

{

nameBox.ToolTip = "";

nameBox.Foreground = Brushes.Black;

}

}

//Проверка пароля

if (password1.Length < 4 && password1.Length > 16)

{

isGood = false;

firstPassBox.ToolTip = "Длина пароля должна быть от 4 до 16 символов!";

firstPassBox.Foreground = Brushes.Red;

}

else

{

firstPassBox.ToolTip = "";

firstPassBox.Foreground = Brushes.Black;

}

//Подтверждение пароля

if (password1 != password2)

{

isGood = false;

secondPassBox.ToolTip = "Пароли не совпадают";

secondPassBox.Foreground = Brushes.Red;

}

else

{

secondPassBox.ToolTip = "";

secondPassBox.Foreground = Brushes.Black;

}

if (isGood == true)

{

if (crt == Criterion.customer)

{

SQLbase.Insert($"insert into Customer(login, name, pass) values (N'{login}',N'{name}',N'{password1}')");

}

if(crt == Criterion.programmer)

{

SQLbase.Insert($"insert into Programmer(login, name, pass) values (N'{login}',N'{name}',N'{password1}')");

}

NavigationService.Navigate(new Uri("/Login.xaml", UriKind.Relative));

}

}

private void ButtonLoginGo(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("/Login.xaml", UriKind.Relative));

}

}

Утверд.

Лист 1

Листов 5

Т.контр.

Провер.

Багласова Е.В.

Н.контр.

ОП Т.895017.401 ГЧ

Программная реализация приложения «Input»

Диаграмма деятельности

У

Масса

Лит.

Масштаб

КБП

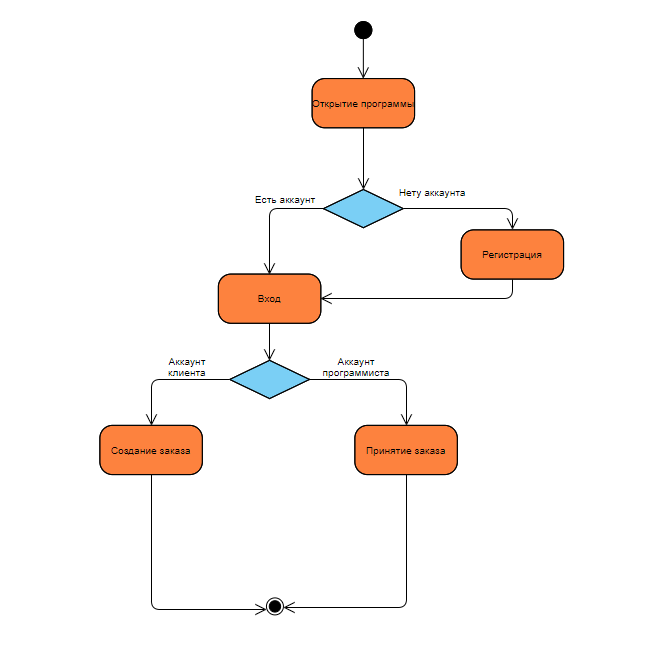
Реценз.

Взам.инв.№

Подп. и дата

Подп. и дата

Инв.№дубл.



№ докум.

Подпись

Дата

Изм.

Лист

Разраб.

Петровский А.А

Инв.№подл.

Утверд.

Лист 2

Листов 5

Т.контр.

Провер.

Багласова Е.В.

Н.контр.

ОП Т.895017.401 ГЧ

Программная реализация приложения «Input»

Диаграмма классов

У

Масса

Лит.

Масштаб

КБП

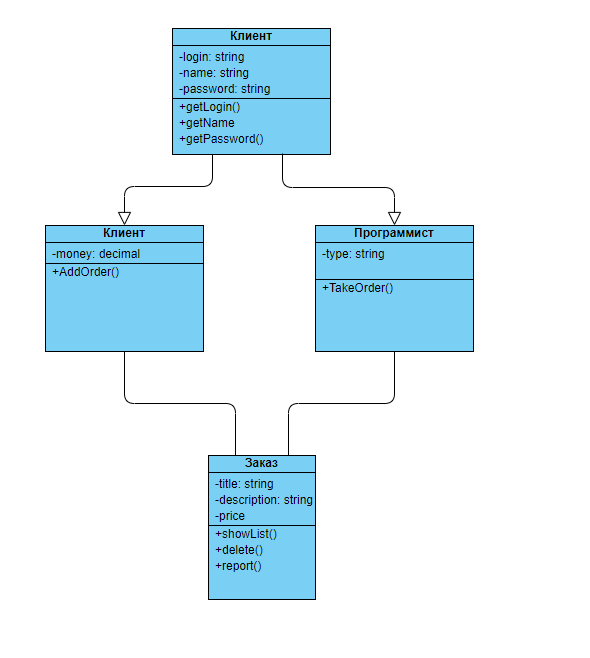
Реценз.

Взам.инв.№

Подп. и дата

Подп. и дата

Инв.№дубл.



№ докум.

Подпись

Дата

Изм.

Лист

Инв.№подл.

Разраб.

Петровский А.А

№ докум.

Подпись

Дата

Изм.

Лист

Утверд.

Лист 3

Листов 5

Т.контр.

Провер.

Багласова Е.В.

Н.контр.

ОП Т.895017.401 ГЧ

Программная реализация приложения «Input»

Диаграмма вариантов использования

У

Масса

Лит.

Масштаб

КБП

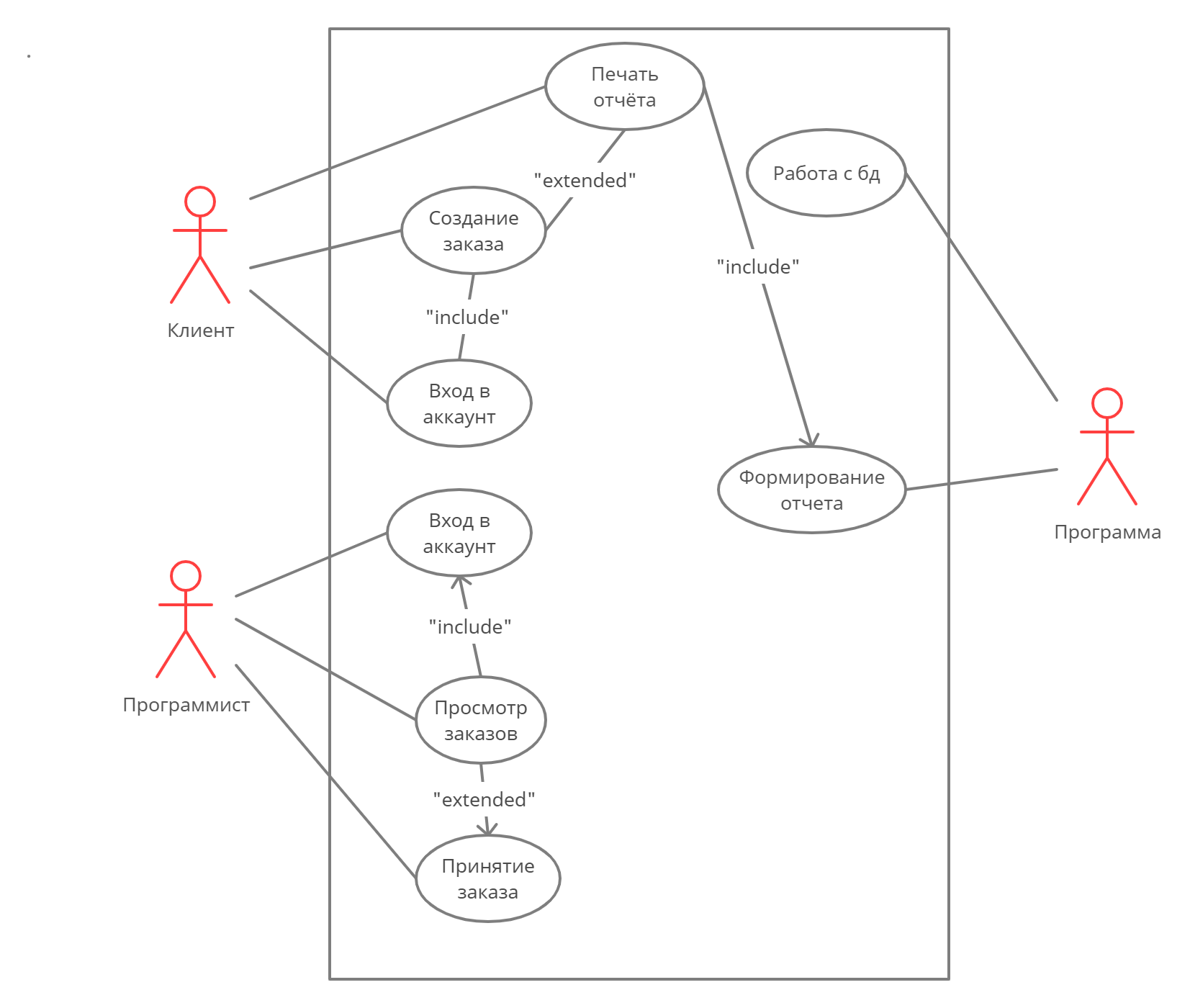
Реценз.

Взам.инв.№

Подп. и дата

Подп. и дата

Инв.№дубл.



№ докум.

Подпись

Дата

Изм.

Лист

Разраб.

Петровский А.А.

Инв.№подл.

№ докум.

Подпись

Дата

Изм.

Лист

Утверд.

Лист 4

Листов 5

Т.контр.

Провер.

Багласова Е.В.

Н.контр.

ОП Т.895017.401 ГЧ

Программная реализация приложения «Input»

Диаграмма компонентов

У

Масса

Лит.

Масштаб

КБП

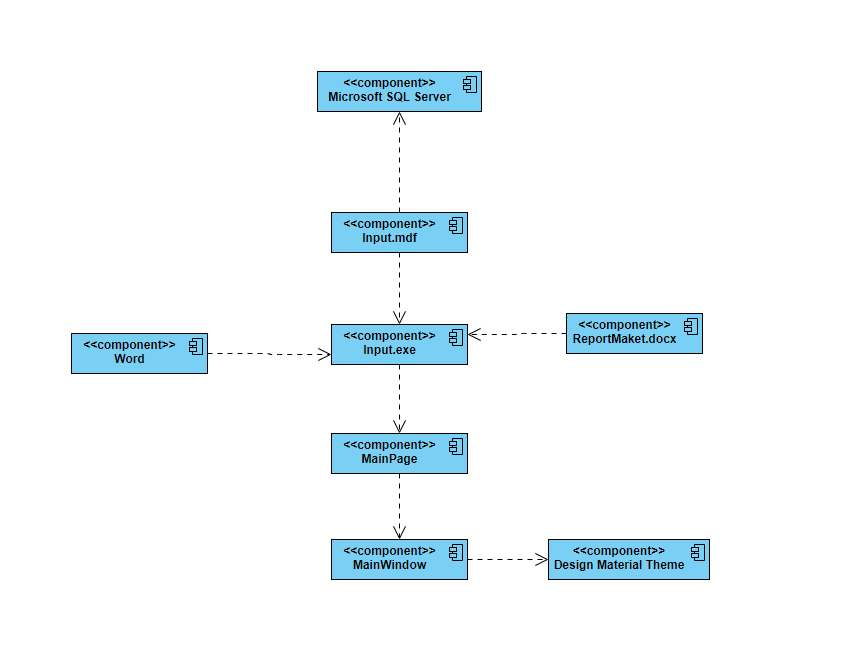
Реценз.

Взам.инв.№

Подп. и дата

Подп. и дата

Инв.№дубл.



№ докум.

Подпись

Дата

Изм.

Лист

Инв.№подл.

№ докум.

Подпись

Дата

Изм.

Лист

Разраб.

Петровский А.А

Утверд.

Лист 5

Листов 5

Т.контр.

Провер.

Багласова Е.В.

Н.контр.

ОП Т.895017.401 ГЧ

Программная реализация приложения «Input»

Диаграмма последовательности

У

Масса

Лит.

Масштаб

КБП

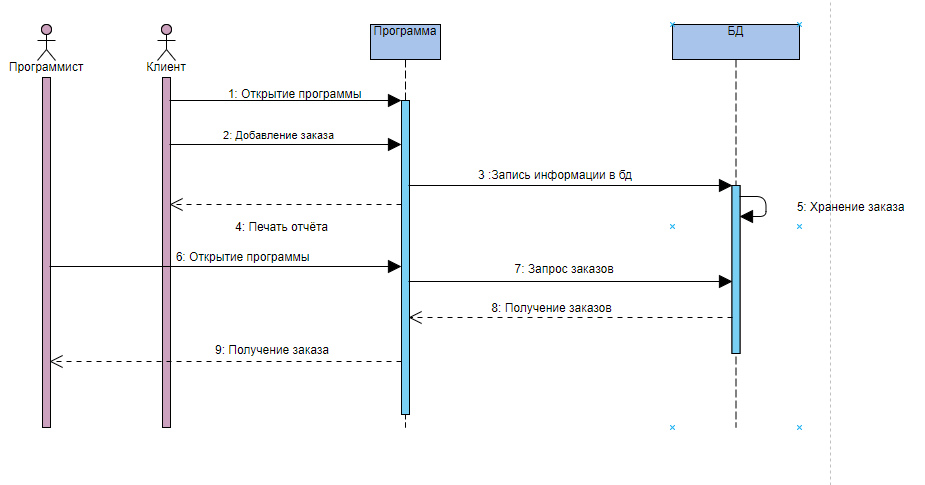
Реценз.

Взам.инв.№

Подп. и дата

Подп. и дата

Инв.№дубл.



№ докум.

Подпись

Дата

Изм.

Лист

Инв.№подл.

Разраб.

Петровский А.А.

№ докум.

Подпись

Дата

Изм.

Лист