МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 6-05-0612-01 Программная инженерия

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Поиск работы»

Исполнитель

студент 2 курса группы 9 Аврусевич Егор Николаевич

(Ф.И.О.)

Руководитель работы ассистент Ромыш А.С.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Смелов В.В.

(подпись)

Минск 2025

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В.Смелов

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 6-05-0612-01 Программная инженерия  Студент: Аврусевич Егор Николавич | Группа:\_\_9\_\_ |
| **Тема: Программное средство «Танцевальная студия»** | |

**1. Срок сдачи студентом законченной работы**: "21 мая 2025 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Модерация предложений о работе.
  + Блокировка пользователя.
  + Добавление, редактирование, удаление предложениями о работе.
* Функции клиента:
  + Выполнять регистрацию и авторизацию.
  + Выбор типа аккаунта.
  + Поиск и фильтрация предложений.
  + Отклик на предложение о работе.
  + Настройка личного кабинета.
  + Размещение предложений о работе.
  + Просматривать информацию о статусе предложения.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

* Введение
* Постановка задачи и обзор литературы
* Проектирование архитектуры проекта
* Разработка функциональной модели и модели данных ПС
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 12.03.2025 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 19.03.2025 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 31.03.2025 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 09.04.2025 |  |
| 5 | Кодирование программного средства | 23.04.2025 |  |
| 6 | Тестирования и отладка программного средства | 07.05.2025 |  |
| 7 | Оформление пояснительной записки | 14.05.2025 |  |
| 9 | Защита проекта | 21.05.2025 |  |

**5. Дата выдачи задания \_\_\_\_**05.03.2025**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *асс. Ромыш А.С.*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Аврусевич Е.Н.*

(дата и подпись студента)

Оглавление

[Введение 5](#_Toc4842)

[1 Аналитический обзор прототипов и литературных источников 6](#_Toc26168)

[1.1 Анализ прототипов 6](#_Toc27042)

[1.1.1 Интернет-ресурс «HeadHunter.ru» 6](#_Toc26901)

[1.1.2 Интернет-ресурс «LinkedIn» 7](#_Toc2199)

[1.1.3 Интернет-ресурс «Rabota.by» 8](#_Toc24499)

[1.2 Постановка задачи 9](#_Toc19433)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc15192)

[2.1 Определение требований к программному средству 10](#_Toc30409)

[2.2 Описание средств разработки 10](#_Toc30060)

[2.3 Диаграмма вариантов использования 11](#_Toc15375)

[3 Проектирование программного средства 12](#_Toc24625)

[3.1 Общая структура приложения 12](#_Toc3225)

[3.2 Взаимоотношение между классами 13](#_Toc31272)

[3.3 Проектирование логической структуры базы данных 13](#_Toc24980)

[3.4 Проектирование архитектуры приложения 17](#_Toc16820)

[3.5 Проектирование архитектуры окон 18](#_Toc10781)

[3.6 Схема работы приложения 20](#_Toc12982)

[4 Реализация программного средства 22](#_Toc32477)

[4.1 Диаграмма классов 22](#_Toc9790)

[4.2 Реализация паттерна MVVM 22](#_Toc5004)

[4.3 Особенности текущей реализации 22](#_Toc19584)

[4.4 Реализация взаимодействия с базой данных 24](#_Toc23932)

[4.4.1 Архитектура доступа к данным 24](#_Toc31660)

[4.4 Реализация методов регистрации и авторизации пользователей 25](#_Toc24482)

[5 Руководство по использованию 27](#_Toc8800)

[Заключение 33](#_Toc20402)

[Список использованных источников 34](#_Toc25290)

[ПРИЛОЖЕНИЕ А 35](#_Toc9542)

[ПРИЛОЖЕНИЕ Б 35](#_Toc17333)

[ПРИЛОЖЕНИЕ В 37](#_Toc27442)

[ПРИЛОЖЕНИЕ Г 38](#_Toc3854)

[ПРИЛОЖЕНИЕ Д 39](#_Toc11371)

# **Введение**

В современном мире цифровые технологии играют ключевую роль в поиске работы. Интернет-ресурсы, мобильные приложения и специализированные программные средства значительно упрощают процесс трудоустройства, помогая соискателям находить вакансии, а работодателям – квалифицированных сотрудников.

Целью данного курсового проекта является разработка программного средства «Поиск работы», которое обеспечит удобный и эффективный процесс поиска вакансий, отклика на них и взаимодействия с работодателями. Разработка подобного программного продукта актуальна в связи с растущей конкуренцией на рынке труда и необходимостью автоматизации поиска подходящих вакансий с учётом пожеланий соискателей.

В рамках проекта будут рассмотрены основные технологии, применяемые при разработке программных средств для трудоустройства, проведен анализ существующих решений и предложены уникальные функциональные возможности, которые сделают программу удобной и востребованной среди пользователей.

Таким образом, данная работа направлена на создание эффективного инструмента, способствующего ускорению и упрощению процесса трудоустройства, а также повышению удобства взаимодействия между работодателями и соискателями.

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами аналогов по данной теме;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

В процессе выполнения курсового проекта будут применены принципы и подходы объектно-ориентированного программирования (ООП), а также будет использована база данных MS SQL Server и технология Windows Presentation Foundation (WPF).

# **1 Аналитический обзор прототипов и литературных источников**

## **Анализ прототипов**

Перед разработкой программного средства необходимо провести анализ существующих решений, выявить их сильные и слабые стороны, а также определить функциональные возможности, которые могут быть улучшены или дополнены. В ходе выполнения курсового проекта были проанализированы цели и задачи, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

### 1.1.1 Интернет-ресурс «HeadHunter.ru»

HeadHunter (https://hh.ru) – одна из крупнейших онлайн-платформ по поиску работы и подбору персонала в России и странах СНГ. Она объединяет миллионы соискателей и работодателей, предлагая удобные инструменты для поиска работы и управления вакансиями. Интерфейс сайта представлен на рисунке 1.1.

|  |
| --- |
|  |

Рисунок 1.1 – Интерфейс сайта hh.ru

На этом сайте пользователь может создать несколько резюме для разных профессий выбрать фильтры (критерии) своей будущей работы, которые ему подходят, а так же посмотреть текущее количество откликнувшихся соискателей на выбранную вакансию.

Главной целью этого сайта является поиск и размещение вакансий о работе. Когда пользователь откликается на вакансию, пользователь сразу отправляет своё резюме работодателю не тратя на это своё время. Работодатель в свою очередь получая резюме сотрудника и просмотрев его может пообщаться с откликнувшимся в чате и если сотрудник ему подходит быстро закрыть вакансию.

Одним из плюсов данного сайта является возможность закрыть своё резюме от нежелательных работодателей.

Сайт неплохой, но имеет свои недостатки. На сайте многие полезные функции платные, а также нет фильтрации откликов кандидатов, многие работодатели часто жалуются на нерелевантные резюме.

## **1.1.2 Интернет-ресурс «LinkedIn»**

Сайт LinkedIn(https://ru.linkedin.com) – это крупнейшая международная социальная сеть для профессионалов, ориентированная на поиск работы, установление деловых связей и развитие карьеры. В отличие от традиционных сайтов вакансий, платформа совмещает в себе функции социальной сети и системы подбора персонала, что делает её уникальным инструментом как для соискателей, так и для работодателей. Интерфейс сайта представлена на рисунке 1.2.

|  |
| --- |
|  |

Рисунок 1.2 – Интерфейс сайта LinkedIn.com

Для соискателей LinkedIn предлагает широкие возможности по созданию профессионального профиля, который служит аналогом резюме. В профиле можно указать образование, навыки, опыт работы, достижения, а также добавить портфолио и сертификаты. Доступна система рекомендаций от коллег и работодателей, что повышает доверие к кандидату.

Поиск работы на платформе реализован с помощью фильтров, позволяющих сортировать вакансии по компании, уровню должности, опыту, местоположению и другим параметрам. LinkedIn также предлагает персонализированные рекомендации вакансий на основе данных профиля пользователя.

Как и сайт hh.ru у платформы есть и недостатки. В первую очередь, LinkedIn ориентирован на квалифицированных специалистов, что затрудняет поиск работы для начинающих. Высокая конкуренция на платформе приводит к тому, что популярные вакансии могут получать сотни откликов, снижая шансы кандидатов на трудоустройство.

## **1.1.3 Интернет-ресурс «Rabota.by»**

Rabota.by(https://rabota.by) одна из ведущих платформ по поиску работы в Беларуси, предоставляющая удобные инструменты как для соискателей, так и для работодателей. Интерфейс сайта представлен на рисунке 1.3, на котором показана главная страница сайта.

|  |
| --- |
|  |

Рисунок 1.3 – Интерфейс сайта Rabota.by

Для соискателей Rabota.by предоставляет возможность создания резюме через удобный конструктор либо загрузки готового документа. Важной особенностью является гибкость в настройках конфиденциальности – пользователь может скрыть своё резюме от определённых компаний. Также можно создать несколько вариантов резюме, если кандидат рассматривает разные профессии. Доступна настройка уведомлений, которые оповещают пользователя о новых подходящих предложениях.

Работодатели могут размещать вакансии как на бесплатной, так и на платной основе, используя дополнительные инструменты продвижения. В их распоряжении большая база резюме, которая позволяет находить кандидатов по ключевым навыкам, опыту работы и региону.

Среди главных преимуществ платформы можно выделить её узкую специализацию на рынке труда Беларуси, что делает вакансии более релевантными для местных соискателей. Кроме того, платформа ориентирована исключительно на Беларусь, поэтому количество вакансий ограничено по сравнению с международными сервисами, что является и своеобразным минусом, количество вакансий существенно уступает международным платформам. Многие специалисты, особенно в сфере IT, не находят достаточного количества вакансий и вынуждены использовать другие сервисы.

Rabota.by отличается современным дизайном, удобной навигацией и высокой скоростью работы.

## **1.2 Постановка задачи**

Обзор аналогов позволяет нам проанализировать свои преимущества и недостатки альтернативных возможностей. Также, благодаря этому, мы можем сформировать список требований для данного программного средства.

Программное средство должно реализовывать следующий функционал:

Функции администратора:

- Модерация предложений о работе.

- Блокировка пользователя.

- Добавление, редактирование, удаление предложениями о работе.

Функции клиента:

- Выполнять регистрацию и авторизацию.

- Выбор типа аккаунта.

- Поиск и фильтрация предложений.

- Отклик на предложение о работе.

- Настройка личного кабинета.

- Размещение предложений о работе.

- Просматривать информацию о статусе предложения.

# **2 Анализ требований к программному средству и разработка функциональных требований**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Основная цель анализа требований в проектах заключается в получении максимально полной информации о клиенте, его задачах и особенностях проекта. Анализ требований помогает определить границы проекта, выявить возможные риски и ключевые требования, как методологические, так и технологические. На этом этапе формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые будут использоваться для оценки результатов внедрения. Определение и описание требований является важным этапом, так как эти требования оказывают влияние на все последующие этапы проекта и определяют его успех.

## 2.1 Определение требований к программному средству

Функционально ПС должно выполнять следующие задачи:

* регистрация и авторизация пользователей;
* сохранение информации в базе данных;
* для администратора добавление/изменение/удаление пользователей, вакансий, откликов на вакансии;
* для пользователя изменение личного кабинета;
* для пользователя просмотр информации о вакансии или соискателе;
* для пользователя(соискателя) отклик на вакансии и просмотр статуса отклика;
* для пользователя(работодателя) добавление/удаление вакансий;
* для пользователя(работодателя) изменения статуса отклика пользователя(соискателя) на его вакансию.

## 2.2 Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Framework;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework;
* SQLite.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Есть различные механизмы работы с БД в С#. Т.к. моя база данных небольшая, мой выбор пал на Entity Framework. Entity Framework предоставляет простой способ доступа к данным в .Net Framework.

2.3 Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является графическим средством моделирования, которое позволяет представить функциональность системы или программного продукта с точки зрения ее пользователей. Она помогает идентифицировать и описать различные сценарии использования системы, включая взаимодействие между актерами (пользователями) и самой системой.

Диаграмма вариантов использования состоит из актеров, вариантов использования (Use Case) и связей между ними. Актеры представляют различных пользователей или внешние системы, которые взаимодействуют с программой. Варианты использования представляют собой функциональные возможности системы, которые могут быть выполнены пользователями или другими актерами.

Преимущества использования диаграммы вариантов использования включают:

Визуализацию функциональности системы: Диаграмма вариантов использования помогает визуализировать все возможные сценарии использования системы, что помогает лучше понять ее функциональность и взаимодействие с пользователями.

Определение требований: Диаграмма вариантов использования помогает идентифицировать и описать функциональные требования к системе на основе сценариев использования.

Улучшение коммуникации: Диаграмма вариантов использования служит средством коммуникации между разработчиками, заказчиками и другими заинтересованными сторонами, позволяя им лучше понять ожидания от данного приложения.

UML-диаграмма вариантов использования для программного средства «HireVich» представлена в приложении А.

3 Проектирование программного средства

3.1 Общая структура приложения

Общая структура проекта Windows Presentation Foundation (WPF) – это организация файлов и папок в рамках проекта WPF. Структура разрабатываемого проекта «HireVich» представлена на рисунке 3.1.

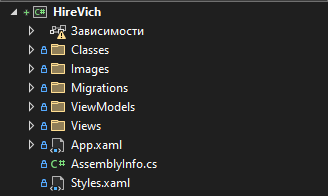


Рисунок 3.1 — Структура проекта

Описание структуры основных папок и файлов проекта и библиотеки классов представлено в таблице 3.1.

Таблица 3.1 — Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| App.xaml | Определение глобальных ресурсов |
| Папка Images | Директория, содержащая подпапки с всем изображениями: иконки, логотипы, изображения профилей пользователей. |
| Папка Classes | Директория, содержащая основные классы данных (или модели), которые описывают структуру объектов, используемых в приложении, и второстепенные классы и функции используемые для автоматизации построения приложения. Эти классы играют роль представлений данных (Data Models) и используются в логике приложения, в интерфейсе и при взаимодействии с базой данных. |
| Styles.xaml | Этот файл определяет внешний вид и поведение элементов управления (например, кнопок, DatePicker, ComboBox), чтобы интерфейс выглядел современно, удобно и последовательно. |

Продолжение таблицы 3.3

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Папка Views | Директория Views содержит несколько окон (Window), папку Pages, в которой находятся страницы (Page) для навигации внутри приложения и папку Controls, в которой находятся ЭУ созданные мной. |
| AssemblyInfo.cs | Файл в котором находится информация компиляции. |
| Папка Migrations | Директория, содержащая классы для создания базы данных и всех необходимых таблиц и зависимостей. |
| Папка ViewModels | Директория, содержащая некоторые класса взаимодействия со страницами. |

Таким образом, сформированная таблица помогает понять общую структура проекта проектируемого программного средства.

**3.2 Взаимоотношение между классами**

Для визуализации взаимосвязей между классами используется диаграмма UML — графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и рёбрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в приложении Б.

**3.3 Проектирование логической структуры базы данных**

Для создания базы данных описываемого приложения использовалась система управления реляционными базами данных DB Browser for SQLite.

База данных — это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных — совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами.

База данных данного курсового проекта состоит из 5 таблиц. Схема базы данных HireVich изображена на рисунке 3.3.

|  |
| --- |
| image_2025-05-17_21-17-49 |

Рисунок 3.3 — Схема базы данных приложения

Остановимся более подробно на некоторых основных таблицах и рассмотрим, какие поля они содержат.

В таблице 3.3 представлена структура таблицы «Users», которая содержит информацию о пользователях системы. Таблица включает в себя персональные данные, а также тип аккаунта.

Таблица 3.3 — Таблица «Users»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| id | INTEGER | Уникальный идентификатор пользователя |
| login | TEXT | Логин пользователя |
| password | TEXT | пароль |
| IsSercher | INTEGER | Хранение типа аккаунта(0 - работодатель, 1 - соискатель) |
| admin | INTEGER | Хранение типа аккаунта(0 - пользователь, 1 - администратор) |

В таблице 3.4 представлена структура таблицы «Users\_data», которая содержит информацию о соискателях. В ней хранятся личная информация, контактные данные, а также фото(в байтах).

Таблица 3.4 — Таблица «Users\_data»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| id | integer | Уникальный идентификатор пользователя |
| FirstName | TEXT | Имя |
| LastName | TEXT | Фамилия |
| Email | TEXT | Электронная почта |
| Age | TEXT | Возрост |
| Country | TEXT | Страна проживания |
| About | TEXT | Резюме |
| Education | TEXT | Образование |
| Photo | BLOB | Фотография |
| Speciality | TEXT | Специальность |

В таблице 3.5 представлена структура таблицы «Companys\_data», которая содержит информацию о работодателях. В ней указаны информация о компании, логотип и контактная информация.

Таблица 3.5 — Таблица «Companys\_data»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | | Описание |
| id | INTEGER | | Уникальный идентификатор компании |
| Discription | TEXT | | Подробная информация о компании |
| Email | | TEXT | Электронная почта |
| Name | | TEXT | Название компании |
| Logo | | BLOB | Логотип |

В таблице 3.6 приведена структура таблицы «Applications», содержащей вакансии на которые нужны работники, в ней сожержится информация о самой вакансии, заработной плате, требуемый опыт и некоторую информацию о работодателе.

Таблица 3.6 — Таблица «Applications»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| id | INTEGER | Уникальный идентификатор вакансии |
| id\_Company | INTEGER | Содержит идентификатор работодателя, опубликовавшего вакансию |
| Company\_name | TEXT | Название компании работодателя |
| Vacation\_Name | TEXT | Название вакансии |
| Wage | INTEGER | Зарплата |
| Description | TEXT | Дополнительная информация о вакансии |
| Country | TEXT | Страна |
| Experience | TEXT | Требуемый опыт |

В таблице 3.7 представлена структура таблицы «Responses», которая содержит информацию о работодателях. В ней указаны информация о компании, логотип и контактная информация.

Таблица 3.7 — Таблица «Responses»

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| id | INTEGER | Уникальный идентификатор компании |
| vacancyId | INTEGER | Индентификатор вакансии |
| aplicantId | INTEGER | Индентификатор соискателя |
| ResponseDate | TEXT | Дата отклика |
| Status | INTEGER | Статус отклика (0 - на рассмотрении, 1 - принят, 2 - непринят) |

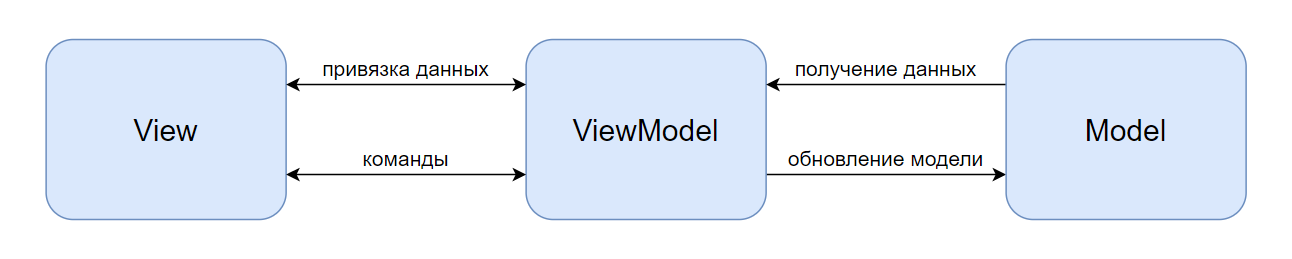
В целом, структура базы данных «Hirevich» предоставляет необходимую основу для хранения и управления информацией в приложении «Hirevich».

## 3.4 Проектирование архитектуры приложения

Архитектура программного обеспечения представляет собой совокупность ключевых решений, касающихся организации программной системы. Она включает выбор структурных элементов и их интерфейсов, определение их взаимодействия, а также объединение этих элементов в более крупные системы. Однако, центральную роль играет архитектурный стиль, который определяет всю организацию системы, включая элементы, их интерфейсы, их сотрудничество и способы их соединения.

Для удовлетворения требований проектируемой системы с точки зрения различных атрибутов качества, применяются различные архитектурные шаблоны, так называемые паттерны. В разрабатываемых приложениях часто используется архитектурный шаблон Model-View-ViewModel (MVVM). Этот шаблон состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

На рисунке 3.4 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

Рисунок 3.4 — Структура шаблона MVVM

Model (модель) описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную с этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления. Часто модель реализует интерфейс INotifyPropertyChanged, который позволяет системе автоматически обнаруживать изменения свойств модели и облегчает привязку к представлению, хотя сама модель не взаимодействует напрямую с представлением.

View (представление) определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление — это код XAML, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. Представление не обрабатывает события, а в основном выполняет действия посредством команд.

ViewModel (модель представления) связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Для своего проекта я выбрал в основу архитектуру Code-Behind, но также использовал архитектуру MVVM для некоторых задач. Code-Behind — это подход в разработке WPF-приложений, при котором логика работы интерфейса пишется в файлах .xaml.cs, связанных с XAML-разметкой. В отличие от MVVM, где используется разделение на Model-View-ViewModel, Code-Behind позволяет напрямую управлять элементами интерфейса и обрабатывать события в коде.

**Преимущества Code-Behind:**

- Быстрое прототипирование — не нужно создавать ViewModel и Binding-команды.

- Понятность — вся логика сосредоточена в одном файле.

- Можно динамически изменять свойства элементов (видимость, текст, стили) без сложных привязок.

- Удобно для анимаций и кастомных взаимодействий.

3.5 Проектирование архитектуры окон

Приложение включает в себя 10 окон и 7 страниц.

При запуске приложения пользователь видит окно авторизации. Пользователь может ввести данные и получить доступ к стандартному функционалу приложения (при входе под ролью обычного пользователя) или же к расширенному функционалу (при входе под ролью администратора), либо перейти к окну «Регистрация», после которого он может ввести данные в форму входа и также получить доступ к приложению.

Внешний вид стартовой страницы «Авторизациии» приложения «Hirevich» представлен на рисунке 3.5.

|  |
| --- |
|  |

Рисунок 3.5 — Стартовая страницы «Hirevich»

Внешний вид страницы регистрации представлен на рисунке 3.6.

|  |
| --- |
|  |

Рисунок 3.6 — Страница регистрации

После входа в приложение, в зависимости от роли, пользователь переходит на страницу администратора, соискателя или работодателя.

Боковое меню состоит из набора кнопок, которые относятся к определённому типу пользователя согласно его уровню доступа. Эти кнопки позволяют перейти на страницы с функционалом. Далее рассмотрены варианты окон.

Страница пользователя(соискателя) даёт доступ к стандартному функционалу приложения для соискателя или работодателя в зависимости от типа аккаунта. Пункты меню страницы изображены на рисунке 3.7, и пункты меню для пользователя(работодателя) на рисунке 3.8.

|  |
| --- |
|  |

Рисунок 3.7 — Пункты меню для пользователя(соискателя)

|  |
| --- |
|  |

Рисунок 3.8 — Пункты меню для пользователя(работодателя)

При авторизации под логином и паролем администратора, можно попасть в панель администратора, пункты меню изображены на рисунке 3.9.

|  |
| --- |
|  |

Рисунок 3.9 — Пункты меню для администратора в панели администратора

Администратор имеет больше технических возможностей, чем остальные пользователи. Об этом свидетельствуют пункты меню, которые позволяют открыть страницы с возможностью просмотра и редактирования таблиц БД.

**3.6 Схема работы приложения**

Для описания маршрутов работы программного продукта была создана диаграмма деятельности, которая показывает пути перехода между страницами приложения, представлена на рисунке 3.10.

|  |
| --- |
|  |

Рисунок 3.10 — Диаграмма деятельности

Для наглядного отображения взаимодействия между объектами системы в различные моменты времени в рамках одного сценария использования применяется диаграмма последовательностей в UML. Эта диаграмма позволяет иллюстрировать, как объекты системы взаимодействуют друг с другом для выполнения определенной функции, а также отображает порядок их взаимодействия в рамках конкретного случая использования.

Пример диаграммы последовательности представлен в приложении на рисунке 3.11.

|  |
| --- |
|  |

Рисунок 3.11 - диаграмма последовательности

Для отображения времени на диаграмме используется линия жизни объекта, которая представляет период его существования в системе и изображается вертикальной штриховой линией, направленной вниз. Объекты представлены прямоугольниками, а сообщения, которыми они обмениваются, обозначаются линиями с стрелками.

**4 Реализация программного средства**

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

**4.1 Диаграмма классов**

В рамках разработки программного решения для организации задач и управления проектами, диаграмма классов является одним из ключевых инструментов моделирования и проектирования системы. Диаграмма классов представляет структуру системы, её классы, атрибуты и отношения между классами.

В процессе проектирования программы для организации задач и управления проектами, диаграмма классов помогает определить основные классы, их свойства и методы. Она позволяет увидеть структуру данных, которые будут использоваться в программе, а также взаимосвязи между классами. Диаграмма классов также помогает лучше понять концептуальную модель системы и обеспечивает основу для дальнейшей разработки.

Диаграмма классов представлена в приложении Б.

**4.2 Реализация паттерна MVVM**

В проекте «Hirevich» архитектурный паттерн MVVM (Model–View–ViewModel) не был реализован. Несмотря на то, что данный подход является рекомендуемым стандартом для построения WPF-приложений, в данной реализации вся логика приложения была сосредоточена непосредственно в code-behind файлов представления, что противоречит концепции разделения ответственности.

Основные причины, по которым MVVM не был реализован:

* Ограниченные сроки выполнения работы — реализация полной MVVM-архитектуры требует дополнительного времени на проектирование моделей, реализацию биндингов, интерфейсов и команд.
* Относительная простота приложения — проект имеет ограниченное количество сущностей и сценариев взаимодействия, что позволило обойтись без абстракции ViewModel.
* Фокус на функциональность, а не архитектуру — приоритет был отдан рабочему функционалу: добавлению, отображению и удалению информации.

## **4.3 Особенности текущей реализации**

В приложении «Hirevich» взаимодействие между пользовательским интерфейсом и бизнес-логикой реализовано напрямую через обработчики событий в code-behind. Такой подход не соответствует архитектуре MVVM, где предусмотрено чёткое разделение: View → ViewModel → Model. Ниже представлены примеры, демонстрирующие отсутствие этого разделения.

В проекте для обработки логики взаимодействия с интерфейсом используется подход Code-behind — вся обработка событий и бизнес-логика реализованы непосредственно в коде страницы (.xaml.cs).

|  |
| --- |
| private void Button\_Click(object sender, RoutedEventArgs e)  {  User? user;  using (var db = new AplicationContext()) {  user = db.Users.FirstOrDefault( u => u.login == login\_box.Text);  if (user != null)  {  if (user.password != password\_box.Password)  {  warning.Visibility = Visibility.Visible;  return;  }  } else {  warning.Visibility = Visibility.Visible;  return;  }  }  MainWindow MainWindow = new MainWindow(user);  MainWindow.Show();  Close();  } |

Листинг 4.1 - Обработка нажатия кнопки “Войти” в окне входа

Такой подход обеспечивает валидацию пользовательского ввода на уровне интерфейса и предотвращает некорректное использование приложения.

|  |
| --- |
| public CVsPage()  {  InitializeComponent();  DataContext = this;  Loaded += OnLoaded;  LoadDataFromDataBase();  var filtersControl = (FiltersPopupCVsControl)FiltersPopupCVs.Child;  filtersControl.ViewModel.FiltersApplied += ApplyFilters;  }  private ObservableCollection<PersonalData> CVs = new ObservableCollection<PersonalData>();  private void LoadDataFromDataBase()  {  try  {  using (var db = new AplicationContext())  {  CVs.Clear();  var CVsFromDb = db.Users\_data.Where(u => db.Users.FirstOrDefault(r => r.id == u.Id).admin == 0).ToList();  foreach (var CV in CVsFromDb)  {  CVs.Add(CV);  }  }  CVsList.ItemsSource = CVs;  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}");  }  } |

Листинг 4.2 - Загрузка данных и привязка к DataGrid

Здесь напрямую вызывается метод DAL-слоя (SQLdata) и устанавливается ItemsSource для DataGrid, что нарушает принцип разделения ответственности.

## 4.4 Реализация взаимодействия с базой данных

Для обеспечения работы приложения была разработана база данных JA.db на основе SQLite. В отличие от PostgreSQL, SQLite - это встраимовая СУБД, не требующая отдельного сервера, что упрощает развёртывание приложения. Структура базы данных соответствует требованиям проекта.

### 4.4.1 Архитектура доступа к данным

Для взаимодествия с базо данных используется Entity Framework Core - современная ORM-технология для .NET, которая предоставляет следующие преимущества:

- Автоматическое создание и обновление схемы БД

- Удобная работа с объектами вместо SQL-запросов

- Поддержка LINQ для составления запросов

- Встроенное кэширование и отслеживание изменения

В листинге 4.3 представлена реализиция класса на основе класса DbContext для взаимодействия с базой данных.

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using System.IO;  namespace JA.Classes  {  public class AplicationContext : DbContext  {public DbSet<User> Users { get; set; }  public DbSet<PersonalData> Users\_data { get; set; }  public DbSet<Companys\_data> Companys\_data { get; set; }  public DbSet<Application> Applications { get; set; }  public DbSet<Response> Responses { get; set; }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {string dbPath = Path.Combine(AppContext.BaseDirectory, "JA.db");  optionsBuilder.UseSqlite($"Data Source={dbPath};Cache=Shared;").LogTo(message => System.Diagnostics.Debug.WriteLine(message));  }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  modelBuilder.Entity<Application>(entity =>  {entity.ToTable("Applications");  entity.HasKey(a => a.Id);  entity.Property(a => a.Id\_Company).HasColumnName("Id\_Company");  });  modelBuilder.Entity<User>()  .HasOne(u => u.Users\_data)  .WithOne(ud => ud.User)  .HasForeignKey<PersonalData>(ud => ud.Id);  modelBuilder.Entity<Companys\_data>()  .HasMany(cd => cd.Applications)  .WithOne(a => a.Companys\_data)  .HasForeignKey(a => a.Id\_Company);  }}} |

Листинг 4.3 - Реализация класса для взаимодествия с базой данных

Преимущества перед ADO.NET:

- Нет необходимости писать SQL-запросы вручную

- Автоматическое управление соединениями

- Встроенная валидация данных

- Поддержка миграций для обновления схемы БД

## **4.4 Реализация методов регистрации и авторизации пользователей**

При запуске приложения пользователь первым делом видит страницу авторизации, в котором располагаются поля для ввода данных и кнопку для перехода к регистрации. В случае ввода корректных данных, он переходит к функционалу приложения, который отличается в зависимости от роли пользователя. Листинг модели представления для авторизации приводится в приложении В.

При отсутствии аккаунта пользователь может зарегистрироваться. Реализация модели представления для регистрации предоставлена в приложении Г.

# 5 Руководство по использованию

При запуске приложения появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «У меня ещё нет аккаунта» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После корректного заполнения всех полей и нажатия кнопки «Зарегистрироваться» появится окно подтверждения регистрации, в котором нужно ввести все дополнительные данные. После успешной регистрации откроется основное окно пользовательского интерфейса.

При наличии аккаунта необходимо ввести корректные данные и войти в аккаунт. Окно авторизации представлено на рисунке 5.1

|  |
| --- |
|  |

Рисунок 5.1 – «Окно авторизации»

После успешной авторизации пользователем откроется окно со страницей «Вакансии» на рисунке 5.2, если это пользователь соискатель или окно “Мои вакансии” на рисунке 5.3, если это пользователь работодатель. Для соискателя отображаются все вакансии, а для работодателя только его вакансии.

|  |
| --- |
|  |

Рисунок 5.2 − Страница – «Вакансии»

Здесь пользователь может откликнуться на любую вакансию или просто просмотреть дополнительную информацию, при нажатии на саму вакансию, высветится окно с информацией о вакансии

|  |
| --- |
|  |

Рисунок 5.3 − Страница – «Мои вакансии»

Переключение между страницами происходит за счёт нажатия на нужную кнопку на верхней панели окна. Помимо кнопок переключения окно на панели есть кнопка “Выйти из аккаунта”.

Следующей страницей является «Личный кабинет», скриншот которого представлен на рисунке 5.4. Информативную часть окна составляют данные пользователя. Функциональная часть это изменение всей информации о пользователе.

|  |
| --- |
|  |

Рисунок 5.4 − Страница – «Личный кабинет»

Далее следует страница «Вакансии» для соискателя, представленная на рисунке 5.5. Здесь при нажатии на вакансию покажется дополнительная информация о само вакансии, а при нажатии на кнопку “откликнуться” вы отправите информацию о себе работодателе, после чего он может решить принимать вас или нет.

|  |
| --- |
|  |

Рисунок 5.5 − Страница – «Вакансии»

Окно «Мои отклики» для соискателя, представленное на рисунке 5.6, содержит в себе все отклики на вакансии, где можно увидеть саму вакансии и статус вашего отклика и если нужно удалить сам отклик.

|  |
| --- |
|  |

Рисунок 5.6 − Страница «Мои отклики»

Следующая страница «Мои вакансии» для работодателя представлена на рисунке 5.7. Здесь работодатель может увидеть вакансии созданные им, а также удалить свою вакансию или создать новую.

|  |
| --- |
|  |

Рисунок 5.7 − Страница «Мои вакансии»

Далее следует страница «Резюме» для работодателя, которая представлена на рисунке 5.8. Работодателю показываются все резюме соискателей, при нажатии на которые он может увидеть дополнительную информацию, в том числе и контактные данные

|  |
| --- |
|  |

Рисунок 5.8 − Страница «Резюме»

Далее следует страница «Мои отклики» для работодателя, которая представлена на рисунке 5.9. Работодателю показываются все отклики соискателей, при нажатии на которые он может увидеть дополнительную информацию об откликнувшемся, посмотрев которое тот может выбрать статус отклика.

|  |
| --- |
|  |

Рисунок 5.9 − Страница «Мои отклики»

# Заключение

В ходе выполнения курсовой работы был разработан проект, являющийся помощником в изучении английского языка. Во время выполнения данной курсовой работы было изучено немало теоретического материала, также просмотрено и разобрано большое количество уже готовых решений тех или иных задач.

В ходе выполнения курсовой работы было выполнено следующее:

* разработка базы данных для хранения информации;
* разработка архитектуры приложения;
* разработка функциональных части приложения;
* разработка пользовательского интерфейса;
* написание исходного кода приложения;
* тестирование приложения.

Проектирование осуществлялось в соответствии с паттерном проектирования code-behind. Для абстрагирования и удобного потенциального расширения приложения были реализованы паттерны работы с данными Repository и Unit of Work. Для разработки дизайна использовалось моё виденье дизайна и словари ресурсов для графического представления данных.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

Само приложение – первый опыт в данной области. Естественно, существует ряд приложений, имеющих функционал и дизайн в разы лучше. Подводя итог всей курсовой работе, можно сделать выводы, что в разработке приложений мне предстоит еще долгий путь.

# Список использованных источников

1. Информационный сайт: Diabetes [Электронный ресурс]. Режим доступа:https://play.google.com/store/apps/details?id=ru.zengalt.simpler&hl=ru&gl=US. Дата доступа: 16.04.2025.
2. Информационный сайт: hh.ru [Электронный ресурс]. Режим доступа:https://hh.ru. Дата доступа: 3.04.2025.
3. Информационный сайт: Rabota.by [Электронный ресурс]. Режим доступа:https://rabota.by. Дата доступа: 3.04.2025.
4. Информационный сайт: LinkedIn [Электронный ресурс]. Режим доступа:https://www.linkedin.com/feed/. Дата доступа: 3.04.2025.
5. Информационный сайт: базы данных [Электронный ресурс]. Режим доступа:[inform.sch117nn.edusite.ru/DswMedia/klyaks\_netuchitelyutestyibazyidannyix.htm](http://inform.sch117nn.edusite.ru/DswMedia/klyaks-_netuchitelyutestyibazyidannyix.htm) – Дата доступа: 3.05.2025.
6. Сайт о программировании Metanit: Введение в Entity Framework [Электронный ресурс]– Режим доступа:https://metanit.com/sharp/efcore/1.1.php – Дата доступа: 29.04.2025.
7. Официальная документация SQLite: SQLite [Электронный ресурс]. Режим доступа:https://www.sqlite.org/docs.html. Дата доступа: 12.04.2025.
8. Официальная документация Microsoft:[Электронный ресурс]. Режим доступа:https://learn.microsoft.com/ru-ru/dotnet/. Дата доступа: 23.04.2025.

**ПРИЛОЖЕНИЕ А**

Диаграмма вариантов использования

|  |
| --- |
|  |

**ПРИЛОЖЕНИЕ Б**

Диаграмма классов для окон (Windows)

|  |
| --- |
|  |

Диаграмма классов для страниц (Pages)

|  |
| --- |
|  |

**ПРИЛОЖЕНИЕ В**

Листинг файла «LoginWindow.xaml»

<Window x:Class="JA.Views.LoginWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:JA"

mc:Ignorable="d"

xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"

Title="Hirevich" Height="444" Width="500" MaxWidth="500" MinWidth="500" MinHeight="444">

<Grid MinWidth="500" MaxWidth="500" Width="495" Margin="0,0,0,-6">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Label Grid.Row="0" Content="Вход" VerticalAlignment="Center" Margin="40,0,0,0" FontSize="35"/>

<Label Grid.Row="1" Content="Login" FontSize="16" Margin="40,0,0,0"/>

<Border Grid.Row="1" CornerRadius="4" HorizontalAlignment="Center" MinWidth="420" Background="#F8F8F8" MaxHeight="48" MinHeight="48">

<TextBox x:Name="login\_box" TextWrapping="Wrap" Text="" Background="#F8F8F8" VerticalAlignment="Center" BorderThickness="0" Margin="10, 0" FontSize="16"/>

</Border>

<Label Grid.Row="2" Content="Пароль" FontSize="16" Margin="40,0,0,0"/>

<Border Grid.Row="2" CornerRadius="4" HorizontalAlignment="Center" MinWidth="420" Background="#F8F8F8" MaxHeight="48" MinHeight="48">

<PasswordBox x:Name="password\_box" Background="#F8F8F8" VerticalAlignment="Center" BorderThickness="0" Margin="10, 0" FontSize="16"/>

</Border>

<Label Grid.Row="2" x:Name="warning" Content="Неверный логин или пароль" Margin="0,75,0,0" Width="170" Foreground="#FFC76666" Visibility="Hidden"/>

<Border Grid.Row="3" CornerRadius="4" HorizontalAlignment="Center" MinWidth="420" Background="#F8F8F8" MaxHeight="48" MinHeight="48"

VerticalAlignment="Top">

<Button BorderThickness="0" Content="Войти" FontSize="16" Click="Button\_Click" />

</Border>

<Button BorderThickness="0" Grid.Row="3" VerticalAlignment="Bottom" HorizontalAlignment="Center" Margin="0,0,0,20" Background="White" FontSize="15" Click="Button\_Click\_1">

<Button.Style>

<Style TargetType="Button">

<Setter Property="Content" Value="У меня ещё нет аккаунта" />

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="White"/>

</Trigger>

</Style.Triggers>

</Style>

</Button.Style>

</Button>

</Grid>

</Window>

**ПРИЛОЖЕНИЕ Г**

Листинг файла «RegisterWindow.xaml»

|  |
| --- |
| <Window x:Class="JA.Views.RegisterWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:JA"  mc:Ignorable="d"  Title="Hirevich" Height="550" Width="500"  MinWidth="500" MinHeight="530"  WindowStartupLocation="CenterScreen">  <Window.Resources>  <!-- Цвета -->  <Color x:Key="PrimaryColor">#3498db</Color>  <Color x:Key="SecondaryColor">#95a5a6</Color>  <Color x:Key="LightGrayColor">#F8F8F8</Color>  <Color x:Key="ErrorColor">#e74c3c</Color>  <SolidColorBrush x:Key="PrimaryBrush" Color="{StaticResource PrimaryColor}"/>  <SolidColorBrush x:Key="SecondaryBrush" Color="{StaticResource SecondaryColor}"/>  <SolidColorBrush x:Key="LightGrayBrush" Color="{StaticResource LightGrayColor}"/>  <SolidColorBrush x:Key="ErrorBrush" Color="{StaticResource ErrorColor}"/>  <SolidColorBrush x:Key="WhiteBrush" Color="White"/>  <SolidColorBrush x:Key="DarkTextBrush" Color="#333333"/>  <SolidColorBrush x:Key="BorderBrush" Color="#C2CDCE"/>  <!-- Стили -->  <Style x:Key="HeaderLabel" TargetType="Label">  <Setter Property="FontSize" Value="32"/>  <Setter Property="FontWeight" Value="SemiBold"/>  <Setter Property="Foreground" Value="{StaticResource DarkTextBrush}"/>  <Setter Property="Margin" Value="40,0,0,20"/>  <Setter Property="VerticalAlignment" Value="Bottom"/>  </Style>  <Style x:Key="InputLabel" TargetType="Label">  <Setter Property="FontSize" Value="16"/>  <Setter Property="Foreground" Value="{StaticResource DarkTextBrush}"/>  <Setter Property="Margin" Value="40,0,0,5"/>  </Style>  <Style x:Key="BaseButtonStyle" TargetType="Button">  <Setter Property="Foreground" Value="{StaticResource DarkTextBrush}"/>  <Setter Property="FontSize" Value="16"/>  <Setter Property="Padding" Value="10"/>  <Setter Property="Cursor" Value="Hand"/>  <Setter Property="BorderThickness" Value="1"/>  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="Button">  <Border x:Name="border"  Background="{TemplateBinding Background}"  BorderBrush="{TemplateBinding BorderBrush}"  BorderThickness="{TemplateBinding BorderThickness}"  CornerRadius="4">  <ContentPresenter HorizontalAlignment="Center"  VerticalAlignment="Center"/>  </Border>  <ControlTemplate.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Setter TargetName="border" Property="Opacity" Value="0.9"/>  </Trigger>  <Trigger Property="IsPressed" Value="True">  <Setter TargetName="border" Property="Opacity" Value="0.8"/>  </Trigger>  </ControlTemplate.Triggers>  </ControlTemplate>  </Setter.Value>  </Setter>  </Style>  <!-- Стиль для активной кнопки -->  <Style x:Key="ActiveToggleButton" BasedOn="{StaticResource BaseButtonStyle}" TargetType="Button">  <Setter Property="Background" Value="{StaticResource LightGrayBrush}"/>  <Setter Property="BorderBrush" Value="{StaticResource SecondaryBrush}"/>  </Style>  <!-- Стиль для неактивной кнопки -->  <Style x:Key="InactiveToggleButton" BasedOn="{StaticResource BaseButtonStyle}" TargetType="Button">  <Setter Property="Background" Value="{StaticResource WhiteBrush}"/>  <Setter Property="BorderBrush" Value="{StaticResource BorderBrush}"/>  <Setter Property="Foreground" Value="{StaticResource BorderBrush}"/>  </Style>  <Style x:Key="InputField" TargetType="Border">  <Setter Property="CornerRadius" Value="4"/>  <Setter Property="Background" Value="{StaticResource LightGrayBrush}"/>  <Setter Property="BorderThickness" Value="1"/>  <Setter Property="BorderBrush" Value="Transparent"/>  <Setter Property="Height" Value="48"/>  <Setter Property="Margin" Value="40,0,40,20"/>  <Setter Property="Padding" Value="10,0"/>  <Style.Triggers>  <Trigger Property="Tag" Value="Error">  <Setter Property="BorderBrush" Value="{StaticResource ErrorBrush}"/>  </Trigger>  </Style.Triggers>  </Style>  <Style x:Key="InputTextBox" TargetType="TextBox">  <Setter Property="Background" Value="Transparent"/>  <Setter Property="BorderThickness" Value="0"/>  <Setter Property="FontSize" Value="16"/>  <Setter Property="VerticalAlignment" Value="Center"/>  </Style>  <Style x:Key="InputPasswordBox" TargetType="PasswordBox">  <Setter Property="Background" Value="Transparent"/>  <Setter Property="BorderThickness" Value="0"/>  <Setter Property="FontSize" Value="16"/>  <Setter Property="VerticalAlignment" Value="Center"/>  </Style>  <Style x:Key="PrimaryButton" TargetType="Button">  <Setter Property="Background" Value="{StaticResource PrimaryBrush}"/>  <Setter Property="Foreground" Value="{StaticResource WhiteBrush}"/>  <Setter Property="BorderThickness" Value="0"/>  <Setter Property="FontSize" Value="16"/>  <Setter Property="Height" Value="48"/>  <Setter Property="Margin" Value="40,0,40,10"/>  <Setter Property="Cursor" Value="Hand"/>  <Setter Property="Template">  <Setter.Value>  <ControlTemplate TargetType="Button">  <Border Background="{TemplateBinding Background}"  CornerRadius="4">  <ContentPresenter HorizontalAlignment="Center"  VerticalAlignment="Center"/>  </Border>  </ControlTemplate>  </Setter.Value>  </Setter>  <Style.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Setter Property="Opacity" Value="0.9"/>  </Trigger>  <Trigger Property="IsPressed" Value="True">  <Setter Property="Opacity" Value="0.8"/>  </Trigger>  </Style.Triggers>  </Style>  <Style x:Key="SecondaryButton" TargetType="Button">  <Setter Property="Background" Value="Transparent"/>  <Setter Property="Foreground" Value="{StaticResource PrimaryBrush}"/>  <Setter Property="BorderThickness" Value="0"/>  <Setter Property="FontSize" Value="15"/>  <Setter Property="Cursor" Value="Hand"/>  <Setter Property="Margin" Value="0,0,0,20"/>  <Style.Triggers>  <Trigger Property="IsMouseOver" Value="True">  <Setter Property="TextBlock.TextDecorations" Value="Underline"/>  </Trigger>  </Style.Triggers>  </Style>  </Window.Resources>  <Grid Background="{StaticResource WhiteBrush}">  <Grid.RowDefinitions>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="Auto"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <!-- Заголовок -->  <Label Grid.Row="0" Content="Регистрация" Style="{StaticResource HeaderLabel}"/>  <!-- Переключатель типа пользователя -->  <Grid Grid.Row="1" Margin="40,0,40,20">  <Grid.ColumnDefinitions>  <ColumnDefinition/>  <ColumnDefinition/>  </Grid.ColumnDefinitions>  <Button x:Name="button\_sercher" Grid.Column="0"  Style="{StaticResource ActiveToggleButton}"  Content="Соискатель"  Click="button\_sercher\_Click"/>  <Button x:Name="button\_company" Grid.Column="1"  Style="{StaticResource InactiveToggleButton}"  Content="Компания"  Click="button\_company\_Click"/>  </Grid>  <!-- Поле логина -->  <StackPanel Grid.Row="2">  <Label Content="Логин" Style="{StaticResource InputLabel}"/>  <Border x:Name="border\_login" Style="{StaticResource InputField}">  <TextBox x:Name="loginTextBox" Style="{StaticResource InputTextBox}"/>  </Border>  <Label x:Name="LoginError" Visibility="Hidden" Style="{StaticResource InputLabel}"/>  </StackPanel>  <!-- Поле пароля -->  <StackPanel Grid.Row="3">  <Label Content="Пароль" Style="{StaticResource InputLabel}"/>  <Border x:Name="border\_password" Style="{StaticResource InputField}">  <PasswordBox x:Name="passwordTextBox" Style="{StaticResource InputPasswordBox}"/>  </Border>  <Label x:Name="PasswordError" Visibility="Hidden" Style="{StaticResource InputLabel}"/>  </StackPanel>  <!-- Кнопка регистрации -->  <Button Grid.Row="4" Content="Зарегистрироваться"  Style="{StaticResource PrimaryButton}"  Click="Button\_Click"/>  <!-- Ссылка на вход -->  <Button Grid.Row="5" VerticalAlignment="Bottom" HorizontalAlignment="Center"  Style="{StaticResource SecondaryButton}"  Content="У меня есть аккаунт"  Click="Button\_Click\_1"/>  </Grid>  </Window> |