МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

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 для обучения алгоритмизации и структурам данных

Исполнитель

студент (ка) 2 курса группы 4 Поздняков Максим Игоревич

(Ф.И.О.)

Руководитель работы ассистент Чистякова Ю. А.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В

(подпись)

Минск 2023

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc135493901)

[1 Аналитический обзор литературы и формирование требований 5](#_Toc135493902)

[1.1 Анализ прототипов 5](#_Toc135493903)

[1.1.1 Metanit 5](#_Toc135493904)

[1.1.2 Программирование-cpp 6](#_Toc135493905)

[1.1.3 Code Lessons 6](#_Toc135493906)

[1.1.4 Википедия 7](#_Toc135493907)

[1.1.5 Школа программирования 8](#_Toc135493908)

[1.2 Аналитический обзор литературных источников 8](#_Toc135493909)

[1.3 Требования к проекту 9](#_Toc135493910)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc135493911)

[2.1 Описание средств разработки 10](#_Toc135493912)

[2.1.1 Microsoft Visual Studio 2022 10](#_Toc135493913)

[2.1.2 Программная платформа .NET 7.0 10](#_Toc135493914)

[2.1.3 Язык программирования C# 11](#_Toc135493915)

[2.1.4 Расширяемый язык разметки XAML 11](#_Toc135493916)

[2.1.5 Технология WPF 11](#_Toc135493917)

[2.1.6 Технология Entity Framework Core 11](#_Toc135493918)

[2.1.7 Microsoft SQL Server 12](#_Toc135493919)

[2.1.8 Протокол SMTP 12](#_Toc135493920)

[2.1.9 Паттерн MVVM 13](#_Toc135493921)

[2.2 Спецификация функциональных требований к программному средству 13](#_Toc135493922)

[2.3 Спецификация функциональных требований 14](#_Toc135493923)

[3 Проектирование программного средства 16](#_Toc135493924)

[3.1 Общая структура 16](#_Toc135493925)

[3.2 Взаимоотношения между классами 24](#_Toc135493926)

[3.3 Модель базы данных 24](#_Toc135493927)

[3.4 Проектирование архитектуры приложения 28](#_Toc135493928)

[3.5 Проектирование последовательностей проекта 28](#_Toc135493929)

[4 Реализация программного средства 30](#_Toc135493930)

[4.1 Основные классы программного средства 30](#_Toc135493931)

[4.2 Описание классов и методов программного средства 30](#_Toc135493932)

[4.2.1 Авторизация 30](#_Toc135493933)

[4.2.2 Регистрация 31](#_Toc135493934)

[4.2.3 Просмотр каталога 31](#_Toc135493935)

[4.2.4 Покупка курса 32](#_Toc135493936)

[4.2.5 Просмотр профиля 32](#_Toc135493937)

[4.2.6 Добавление вопроса 32](#_Toc135493938)

[4.2.7 Приложение администратора 32](#_Toc135493939)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 34](#_Toc135493940)

[5.1 Тестирование авторизации и регистрации 34](#_Toc135493941)

[5.2 Тестирование оплаты книги 37](#_Toc135493942)

[5.3 Тестирование приложения администратора 39](#_Toc135493943)

[6 Руководство по установке и использованию 40](#_Toc135493944)

[ЗАКЛЮЧЕНИЕ 44](#_Toc135493945)

[Список использованных источников 45](#_Toc135493946)

[ПРИЛОЖЕНИЕ А 46](#_Toc135493947)

[ПРИЛОЖЕНИЕ Б 47](#_Toc135493948)

[ПРИЛОЖЕНИЕ В 48](#_Toc135493949)

# ВВЕДЕНИЕ

Программное средство для обучения алгоритмизации и структурам данных может помочь новичкам в программировании освоить базовые концепции, связанные с алгоритмами и структурами данных, такие как сортировки, стек, очередь, списки, поиск, графы и так далее. Программное средство может быть использовано в образовательных целях, как для студентов, так и для преподавателей. Оно может помочь студентам улучшить свои знания по алгоритмам и структурам данных и дать преподавателям дополнительный инструмент для преподавания этих тем. Также его можно использовать как справочник, к которому можно обратиться в любой момент, что повторить уже изученные алгоритмы и структуры данных.

Цель данного курсового проекта - разработать программное средство с реализацией перехода пользователя на новый уровень с помощью тестов, осуществить возможность скачивания презентаций. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы объектно-ориентированного программирования. Также будут использоваться технологии Windows Presentation Foundation(WPF) и база данных Microsoft SQL Server.

В рамках курсового проекта будут рассматривать и реализовываться такие функциональные возможности, как скачивание презентаций, выполнение тестов, покупка курсов, возможность оставления вопросов, удобное управление вкладками.

# 1 Аналитический обзор литературы и формирование требований

## 1.1 Анализ прототипов

Был проведен анализ целей и задач, поставленных в рамках данного курсового проекта, а также изучены примеры аналогичных решений. Исходя из этого анализа были определены требования к разрабатываемому программному средству, учитывающие как все его преимущества, так и недостатки рассмотренных альтернативных решений.

### 1.1.1 Metanit

Наиболее популярное альтернативное решение — интернет-ресурс «Metanit».

Интернет-ресурс «Metanit» представляет собой платформу для обучения программированию на разных языках. Анализируя данный ресурс, можно выделить следующие преимущества: удобный интерфейс, широкий выбор тем, простой и приятный дизайн. Также можно отнести к преимуществам добавление блоков с кодом и результатом работы. Пользователи данного ресурса могут оставлять вопросы под каждой темой, на которые могут отвечать администраторы или другие пользователи.

К недостаткам можно отнести следующие моменты: полностью отсутствует раздел с алгоритмизацией, нет возможности проверить свои знания.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

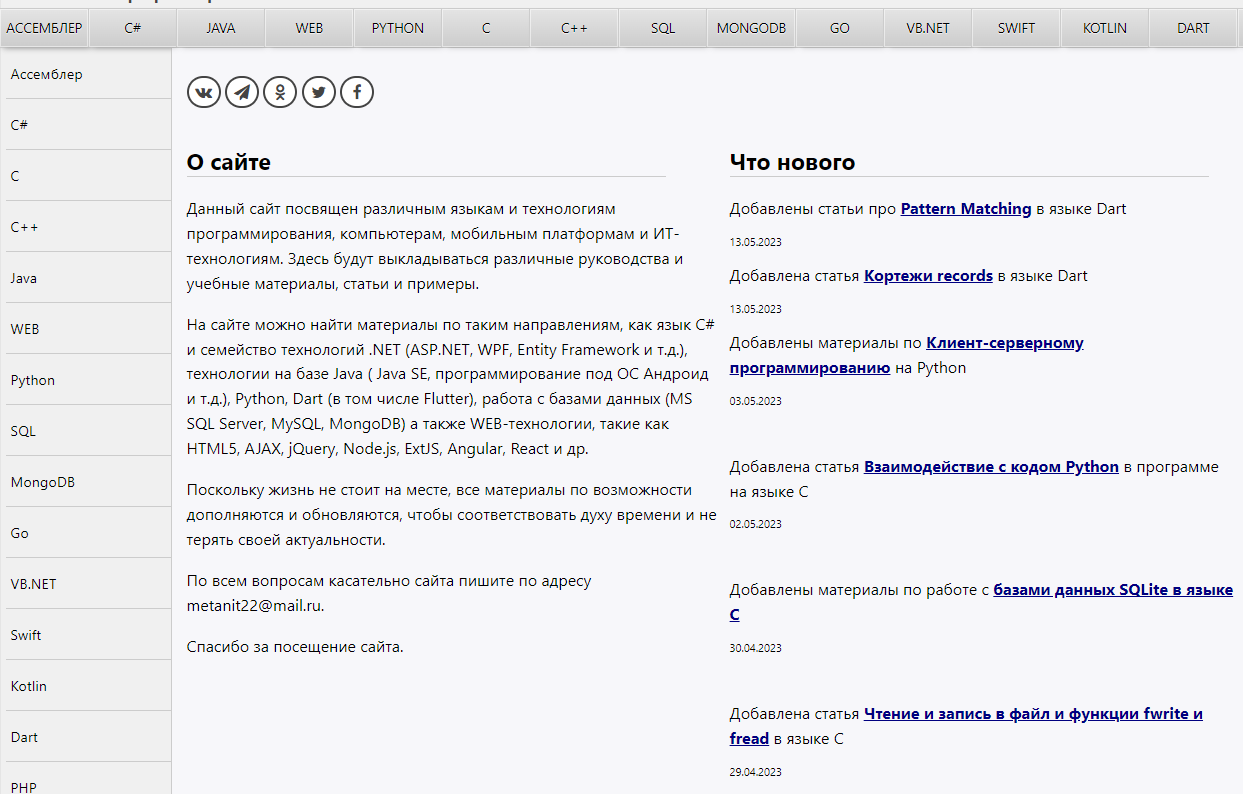


Рисунок 1.1 – Интернет-ресурс «Metanit»

Данный ресурс использует минималистичный дизайн, где основной акцент делается на контенте. Минималистический подход позволяет пользователям сосредоточиться на информации, не отвлекаясь на излишние элементы интерфейса.

### 1.1.2 Программирование-cpp

К наиболее популярным альтернативным решениям также можно отнести приложение от Google Inc под названием «Программирование-cpp».

Программирование-cpp - это платформа обучения языку программирования C++, базовым алгоритмам и структурам данных. Из плюсов можно выделить вставленные блоки кода на языке C++ к примерам алгоритмов, рисунки с наглядным изображением работы алгоритма, возможность оставлять вопросы.

Но, не смотря на все преимущества данного приложения, в нем также можно выделить следующие недостатки: редкое обновление, новые статьи появляются крайне редко, много рекламы, затрудняющей чтение.

Интерфейс приложения представлен на рисунке 1.2.

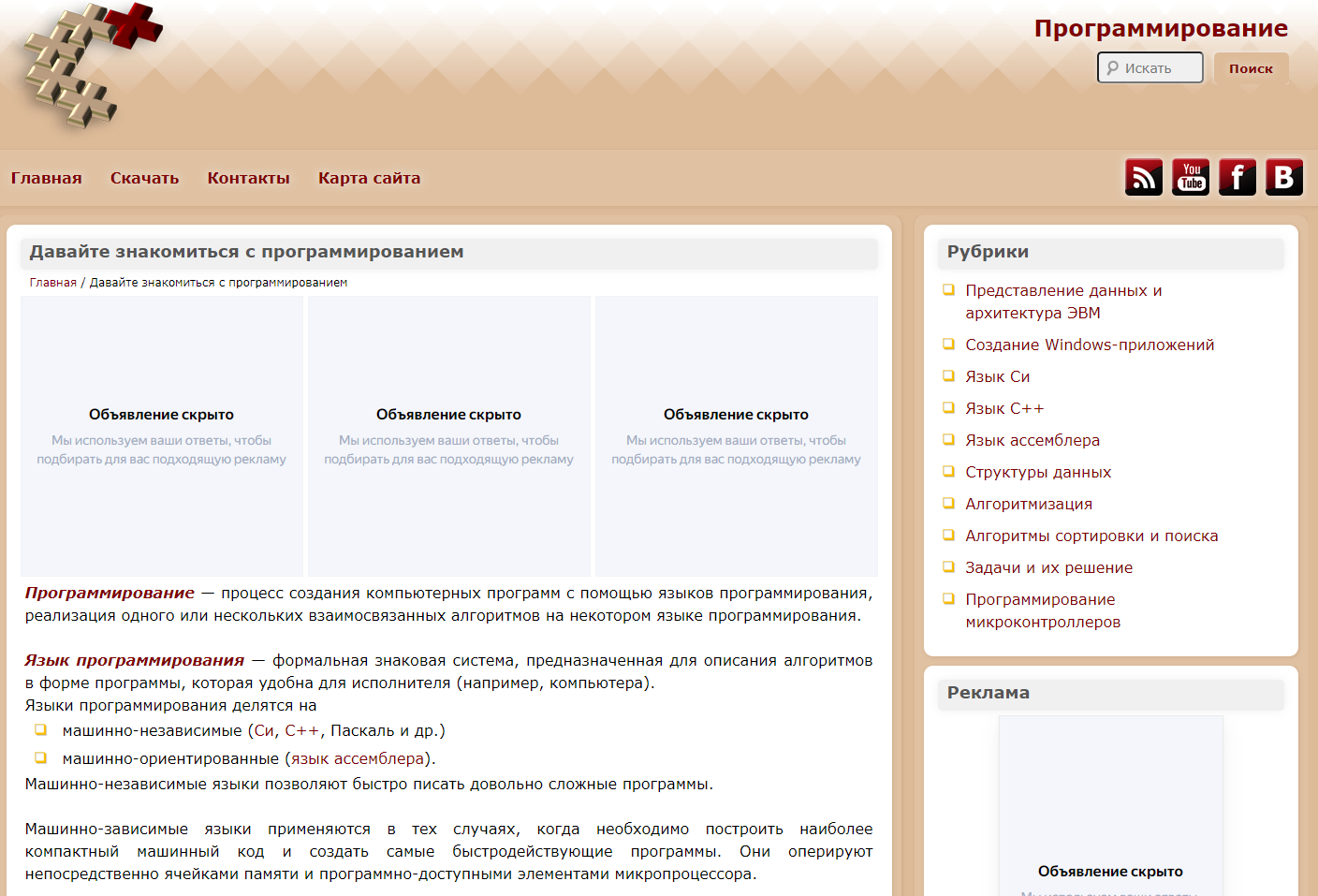


Рисунок 1.2 – Приложение «Google Play Книги»

Дизайн Программирование-cpp отличается простотой. Дизайн Программирование-cpp использует принципы блочного дизайна, которые придают интерфейсу современный и стильный вид.

### 1.1.3 Code Lessons

Ещё одной альтернативой решения поставленных задач является интернет-ресурс «Code Lessons».

Code Lessons - это онлайн-платформа для обучения алгоритмизации и структурам на языке программирования C++. Данный сайт имеет статьи на самые базовые алгоритмы и структуры данных, поэтому он пойдет новичкам. Помимо языка C++, также имеются статьи про язык Python.

Также были выделены следующие недостатки: ограниченный выбор тем, ресурс нагроможден рекламой, крайне редко обновляем

.

Интерфейс интернет-ресурса «Code Lessons» представлен на рисунке 1.3.

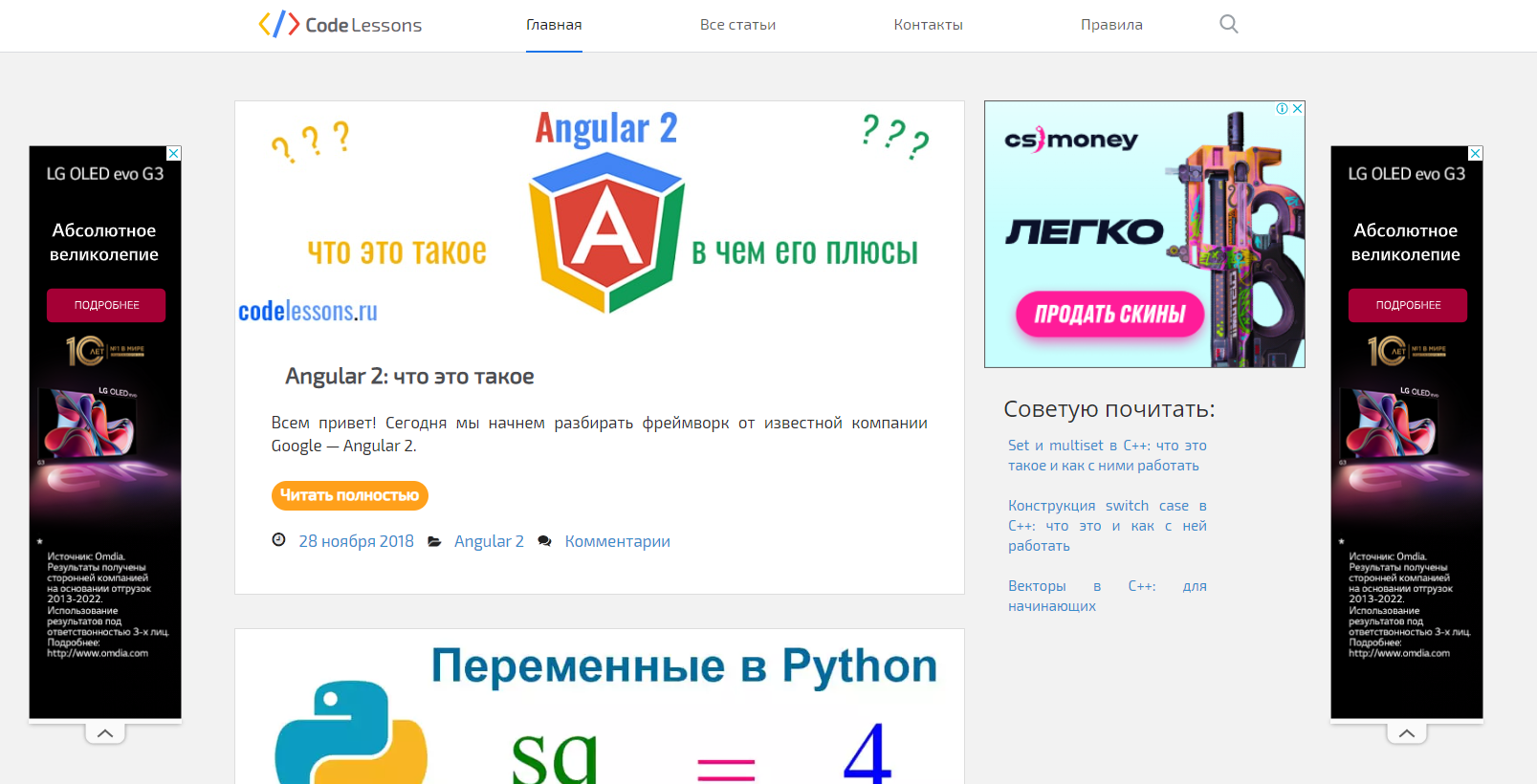


Рисунок 1.3 – Интернет-ресурс «Code Lessons»

Дизайн LiveLib представляет собой современную и привлекательную визуальную концепцию, созданную с учетом потребностей пользователей в удобстве и функциональности. LiveLib использует яркую цветовую палитру, которая создает энергичную и динамичную атмосферу на сайте. Цвета акцентов помогают выделить важные элементы интерфейса и привлекают внимание пользователей.

### 1.1.4 Википедия

В список рассматриваемых альтернатив решения поставленных задач также можно включить интернет-ресурс «Википедия».

Википедия (или Wikipedia) — это всемирная онлайн-платформа, предназначенная для обмена и публикации материала на любую тему. Контент на сайте организован в виде статей и категорий, которые позволяют пользователю быстро найти необходимую информацию. Статьи содержат обширную и полезную информацию, и сайт предоставляет ссылки на другие статьи, связанные с текущей темой.

Наряду со всеми преимуществами были выделены следующие недостатки: нет возможности оставлять вопросы, хоть контент на темы программирования достаточно обширный и содержит информацию на множество тем, однако, данная информация не является исчерпывающей и может служить лишь вводным материалом для изучения этих тем, так как этот сайт не специализируется на обучении программированию.

Интерфейс интернет-ресурса «Википедия» представлен на рисунке 1.4.

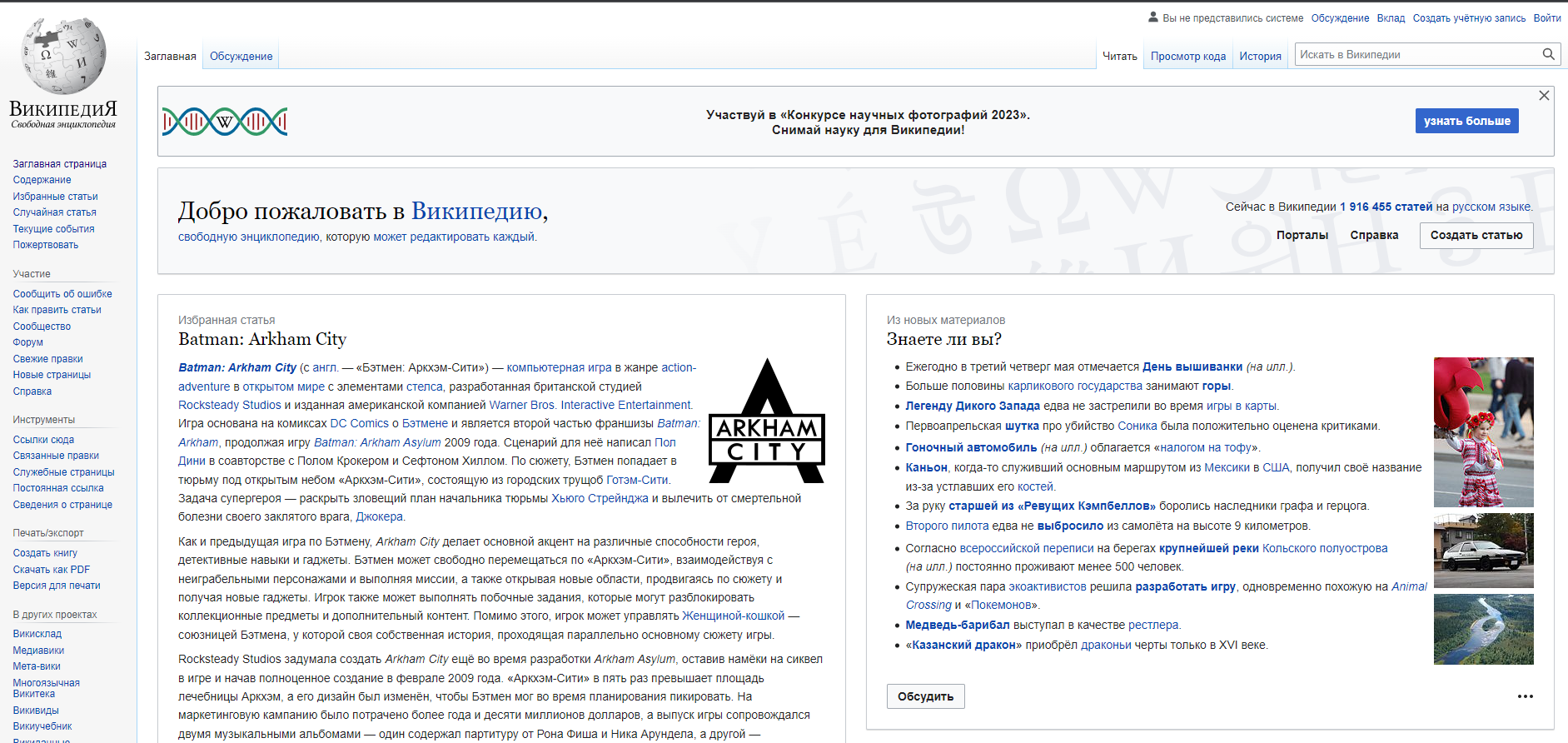


Рисунок 1.4 – Интернет-ресурс «Википедия»

Дизайн сайта довольно стандартный и простой, с использованием основных цветов, таких как белый, серый. Это дает сайту чистый и простой визуальный облик, что позволяет пользователю сосредоточиться на контенте и не отвлекаться на слишком яркие или громоздкие элементы дизайна. Шрифты читаемые и размеры текста сбалансированы.

### 1.1.5 Школа программирования

Одной из возможных альтернатив также является посещение учреждений, в которых проводят лекции для обучения программированию и дают домашнее задание.

Анализируя данный вариант, были выделены следующие преимущества: возможность работать в команде над общим проектом, структурированное обучение, подходит всем.

Но одним из самых важных преимуществ данного выбора является личный контакт с преподавателем, возможность задать ему вопрос напрямую и сразу же услышать ответ.

Исходя из анализа были выделены следующие недостатки: обучение в фиксированное время, большая стоимость обучения, которая зачастую может не соответствовать качеству преподаваемого материала.

## 1.2 Аналитический обзор литературных источников

В процессе разработки программного средства для обучения алгоритмизации с структурам данных была проведена обширная литературная работа, включающая изучение специализированной технической литературы, учебно-методических пособий и справочных материалов, а также анализ статей и публикаций, доступных в интернете.

Можно выделить следующий источник: официальная документация Microsoft, которая предоставляет подробную информацию о различных аспектах WPF, также предоставляет различную информацию о различных аспектах работы с базой данных. Дополнительная информация о принципах работы WPF и баз данных была получена из различных статей интернет-источников, таких как Metanit.

## 1.3 Требования к проекту

В ходе анализа вышеперечисленных аналогов и ознакомления с различными литературными источниками были выделены преимущества и недостатки альтернативных решений, что позволяет корректно сформулировать список требований, которые должны быть соблюдены при разработке программного средства. Эти требования служат основой для проектирования, разработки и тестирования системы.

Программное средство должно обеспечивать выполнение следующих функциональных требований:

* регистрация и аутентификация: пользователь должен иметь возможность создать учетную запись в системе и аутентифицироваться для доступа к функционалу приложения;
* скачивание презентаций: пользователь должен иметь возможность скачивать презентации из приложения;
* вопросы: пользователь должен иметь возможность оставлять вопросы администрации;
* управление пользовательским аккаунтом: пользователь должен иметь возможность изменять фото в профиле;
* произведение оплаты: пользователь должен иметь возможность производить оплату курсов и ожидать сообщения об успешной оплате на электронную почту.

Нефункциональные требования:

* интерфейс пользователя: интерфейс должен быть интуитивно понятным и удобным для использования, с хорошим пользовательским опытом;
* безопасность: система должна обеспечивать безопасность пользовательских данных, включая защиту от несанкционированного доступа.

# 2 Анализ требований к программному средству и разработка функциональных требований

## 2.1 Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET 7.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* MS SQL Server;
* протокол SMTP;

Все перечисленные выше инструменты помогут разработать приложения опираясь на все поставленные требования к приложению.

### 2.1.1 Microsoft Visual Studio 2022

Microsoft Visual Studio - это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Она предоставляет разработчикам широкий спектр инструментов и функциональности для создания программного обеспечения под различные платформы, включая Windows, macOS, iOS, Android и веб-приложения.

Visual Studio также предлагает возможности для создания графического интерфейса приложений с использованием Windows Presentation Foundation (WPF), Windows Forms и других технологий. Она поддерживает создание веб-приложений с использованием ASP.NET, разработку мобильных приложений для платформы Xamarin, а также разработку облачных приложений на базе Microsoft Azure.

### 2.1.2 Программная платформа .NET 7.0

Программная платформа .NET 7 (или .NET Core 7) является одной из версий платформы .NET, разработанной компанией Microsoft. .NET является средой выполнения и набором инструментов для разработки и выполнения приложений под различные операционные системы, включая Windows, macOS и Linux.

.NET 7 представляет собой новую версию платформы, в которой внедрены различные улучшения и новые возможности. Она обеспечивает разработчикам возможность создавать масштабируемые, быстрые и безопасные приложения с использованием различных языков программирования, таких как C#, Visual Basic, F# и другие.

Одним из главных преимуществ платформы .NET 7 является ее инновационность и способность следовать современным трендам в разработке программного обеспечения.

### 2.1.3 Язык программирования C#

За основу языка программирования в приложении взят C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он является одним из основных языков разработки приложений для платформы .NET. C# сочетает в себе элементы языков C и C++, но имеет более простой синтаксис и включает удобные средства для разработки приложений.

C# широко используется для разработки приложений на платформе .NET, включая приложения для Windows, веб-приложения, игры, мобильные приложения и многое другое. Он остается одним из самых популярных языков программирования благодаря своей простоте, мощности и широкой поддержке.

### 2.1.4 Расширяемый язык разметки XAML

XAML (Extensible Application Markup Language) - это язык разметки, используемый в технологии WPF (Windows Presentation Foundation) для описания структуры и внешнего вида элементов пользовательского интерфейса (UI) в декларативной форме. XAML является основным инструментом для создания пользовательского интерфейса в WPF.

XAML используется не только в WPF, но также в других технологиях, таких как Universal Windows Platform (UWP) и Xamarin.Forms, для описания пользовательского интерфейса.

### 2.1.5 Технология WPF

WPF (Windows Presentation Foundation) - это технология разработки графического интерфейса пользователя (GUI) для приложений под управлением операционной системы Windows. Она была разработана компанией Microsoft и является одной из основных частей платформы .NET. WPF предоставляет разработчикам мощные инструменты и возможности для создания современных и привлекательных пользовательских интерфейсов.

WPF использует язык разметки XAML (Extensible Application Markup Language), который позволяет описывать структуру и внешний вид элементов интерфейса в декларативной форме. Это упрощает процесс разработки и позволяет разработчикам и дизайнерам эффективно сотрудничать.

WPF применяет паттерн MVVM (Model-View-ViewModel), который способствует разделению логики приложения от представления. Это позволяет создавать более гибкие и тестируемые приложения.

### 2.1.6 Технология Entity Framework Core

Entity Framework Core (EF Core) - это современная технология доступа к данным в .NET, предоставляющая ORM (Object-Relational Mapping) функциональность для работы с реляционными базами данных. EF Core является легковесной и переносимой версией Entity Framework, оптимизированной для работы с платформой .NET Core и .NET 5+.

Entity Framework Core представляет собой мощный инструмент для работы с данными в .NET, упрощающий и ускоряющий разработку приложений, связанных с базами данных.

### 2.1.7 Microsoft SQL Server

Microsoft SQL Server (MS SQL Server) - это реляционная система управления базами данных (СУБД), разработанная компанией Microsoft. Она предоставляет средства для хранения, организации и обработки структурированных данных, которые могут быть использованы в приложениях и веб-сайтах. MS SQL Server базируется на реляционной модели данных, где данные организованы в таблицы с определенными связями и ограничениями целостности. Это позволяет эффективно хранить и извлекать данные.

MS SQL Server является одной из наиболее популярных реляционных СУБД и широко применяется в различных сферах, включая предприятия, веб-разработку, аналитику данных и многое другое.

### 2.1.8 Протокол SMTP

SMTP (Simple Mail Transfer Protocol) - это стандартный протокол, используемый для отправки и доставки электронной почты через сети TCP/IP. Он предоставляет набор правил и команд для передачи электронных писем между почтовыми серверами.

SMTP обеспечивает надежную доставку электронной почты, выполняя следующие функции:

* отправка почты: SMTP клиент отправляет электронное письмо на SMTP сервер. Он устанавливает соединение с сервером, а затем передает письмо в соответствии с протоколом SMTP;
* маршрутизация: SMTP серверы маршрутизируют письма к целевым серверам, основываясь на информации о доменах получателя и настройках DNS (системы доменных имен);
* проверка подлинности: SMTP серверы могут использовать различные методы проверки подлинности отправителя, чтобы предотвратить спам и несанкционированную отправку писем;
* очередь доставки: если целевой сервер недоступен или временно недоступен, SMTP сервер может сохранить письмо в очереди доставки и повторить попытку доставки позже.

SMTP является одним из основных протоколов, используемых для отправки и доставки электронной почты в Интернете. Он работает совместно с другими протоколами, такими как POP3 (Post Office Protocol version 3) и IMAP (Internet Message Access Protocol), которые используются для получения писем с почтового сервера.

### 2.1.9 Паттерн MVVM

MVVM (Model-View-ViewModel) - это шаблон проектирования, который используется в разработке программного обеспечения, особенно в контексте пользовательского интерфейса. Он разделяет компоненты приложения на три основных уровня: модель (Model), представление (View) и модель представления (ViewModel). Каждый уровень выполняет свои функции и имеет свою ответственность:

* модель (Model): это слой данных, который представляет бизнес-логику, хранение данных и взаимодействие с базой данных или внешними источниками данных. Модель содержит объекты данных, методы для получения и сохранения данных, а также логику обработки данных;
* представление (View): это слой пользовательского интерфейса, который отображает данные пользователю и обрабатывает пользовательские взаимодействия. Представление предоставляет пользовательский интерфейс, который визуализирует данные из модели и позволяет пользователю взаимодействовать с ними;
* модель представления (ViewModel): это слой, который связывает модель и представление. Модель представления предоставляет данные и логику, необходимую для отображения данных в представлении и обработки пользовательских взаимодействий. Он предоставляет свойства и команды, которые связываются с элементами пользовательского интерфейса в представлении, и обновляет модель при необходимости.

Основная идея MVVM заключается в разделении логики приложения и пользовательского интерфейса. Модель предоставляет данные и функциональность, представление отображает эти данные пользователю, а модель представления обеспечивает связь между моделью и представлением. Это позволяет улучшить разделение обязанностей, повысить переиспользуемость кода и упростить тестирование.

## 2.2 Спецификация функциональных требований к программному средству

Спецификация функциональных требований к программному средству для обучения алгоритмизации и структурам данных описывает основные функции и возможности, которые предоставляются пользователям и администраторам системы.

Были поставлены следующие функциональные требования к программному средству:

Для пользователя:

* регистрация;
* авторизация;
* просмотр каталога алгоритмов;
* скачивание отдельных алгоритмов;
* просмотр каталога тестов;
* просмотр каталога курсов;
* возможность покупки курса;
* получение информации о покупке курса на почту;
* возможность задавать вопросы администратору;
* просмотр собственного профиля;
* изменение профиля;

Для администратора:

* авторизация;
* ответ на вопрос пользователя путем отправки сообщений на почту;
* просмотр каталога алгоритмов;
* изменение, удаление и добавление алгоритмов;
* просмотр каталога тестов;
* изменение, удаление и добавление тестов;
* просмотр каталога курсов;
* изменение, удаление и добавление курсов;
* возможность повысить уровень пользователя;

В целом, спецификация функциональных требований является важным этапом в разработке программного средства, поскольку она позволяет четко определить цели и задачи проекта, установить требования пользователей и обеспечить успешную реализацию программного продукта.

Таким образом были представлены основные функциональные требования к разрабатываемому программному средству для обучения алгоритмизации и структурам данных.

## 2.3 Спецификация функциональных требований

Для обеспечения функциональности приложение необходимо правильное проектирование базы данных и ее создание. Подробнее база данных описана в разделах ниже.

При регистрации пользователь должен указывать логин, электронную почту и пароль. Если данные при регистрации пройдут валидацию, то пользователь успешно зарегистрируется. При аутентификации вводится логин и пароль.

Пользователь должен просматривать каталог алгоритмов, тестов и курсов, оставлять вопросы, поэтому все эти данные хранятся в таблицах базах данных, как и информация об покупке курсов. Для каждого пользователя создается собственный профиль, в котором он может изменить фото.

Администратор обладает теми же возможностями, что и пользователь, но он может просматривать, изменять и удалять определенные алгоритмы, тесты, курсы, отвечать на вопросы пользователей путем отправки сообщения на почту. Также может повысить уровень пользователя исходя из результатов тестирования.

Описание функциональности программного средства для обучения алгоритмизации и программирования представлено на UML-схеме. UML-схема представлена в приложении Б.

Система разграничения прав пользователей, включая администратора, обеспечивает эффективный контроль над доступом и ограничение возможностей использования программного средства в соответствии с установленными требованиями и политиками. Это позволяет точно определить, какие функции и ресурсы могут быть доступны для каждой категории пользователей и предотвратить несанкционированный доступ или использование функциональности, которая не соответствует их роли или полномочиям. Это важный аспект создания надежной и защищенной среды для работы с программным средством.

# 3 Проектирование программного средства

## 3.1 Общая структура

Программное средство для обучения алгоритмизации и программирования имеет следующею структуру, представленную на рисунке 3.1.

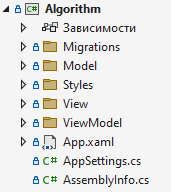


Рисунок 3.1 – Структура проекта

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

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Migrations | Папка, которая создается автоматически при миграции с базой данных. |
| Папка Model | Здесь описаны модели, с которыми происходит вся работа в приложении:   * клиент; * пользователь; * база данных; * объекты для работы с БД. |
| Папка View | В этой папке хранятся файлы с xaml-разметкой, то есть окна, страницы. |
| Папка ViewModel | В данной папке хранятся все классы для взаимодействия графической составляющей и бизнес логики приложения. |
| App.xaml | Файл, определяющий общие ресурсы приложения. |
| AssemblyInfo.cs | Файл содержит информацию о сборке (assembly), такую как версия, описание, автор и другие атрибуты. |

Окончание таблицы 3.1

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| AppSettings.cs | Файл содержит статический класс, который содержит в себе информацию о текущем авторизовавшимся пользователе в приложении. |

Подробная структура папки Migrations представлена на рисунке 3.2.



Рисунок 3.2 — Подробная структура папки «Migrations»

Описание файлов представлено в таблице 3.2.

Таблица 3.2 – Описание файлов папки «Migrations»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Файл 20230518192020\_Initial.cs | Файл миграции, который содержит код на языке программирования (обычно на C#), описывающий изменения, которые должны быть применены к базе данных для создания начальной схемы. |
| Файл ApplicationContextModelSnapshot.cs | Файл содержит информацию о текущей структуре базы данных, включая таблицы, поля и связи. |

Миграция в Entity Framework Core является механизмом, который позволяет управлять изменениями в базе данных в процессе разработки приложения. Она позволяет автоматически создавать, обновлять и откатывать схему базы данных, чтобы она соответствовала моделям данных приложения.

Подробная структура папки Model представлена на рисунке 3.3.

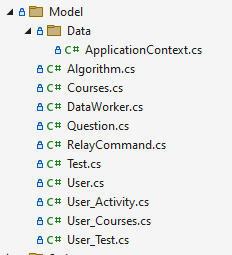


Рисунок 3.3 — Подробная структура папки «Model»

Описание файлов и внутренних папок папки представлено в таблице 3.3.

Таблица 3.3 – Описание файлов и внутренних папок папки «Model»

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Папка Data | Папка хранит в себе класс для работы с базой данных. |
| Файл ApplicationContext.cs | Файл предназначен для подключения к базе данных и использования всех таблиц, который присутствуют в данной базе данных. |
| Файл Algorithm.cs | Данный файл содержит класс Algorithm, который используется для хранения данных об алгоритмах. |
| Файл Courses.cs | Данный файл содержит класс Courses, который используется для хранения данных о курсах. |
| Файл DataWorker.cs | Данный файл содержит класс DataWorkerAdminApp, который содержит функции для работы с базой данных в приложении. |
| Файл Question.cs | Данный файл содержит класс Question, который используется для хранения данных о вопросах. |
| Файл RelayCommand.cs | Файл содержит класс RelayCommand, который реализует интерфейс ICommand, для реализации паттерна Command. |

Окончание таблицы 3.3

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл Test.cs | Данный файл содержит класс Test, который предназначен для хранения в себе информации о тестах. |
| Файл User.cs | Данный файл содержит класс User, который предназначен для хранения в себе информации о пользователях. |
| Файл User\_Activity.cs | Данный файл содержит класс User\_Activity, который предназначен для хранения в себе информации о пройденных алгоритмах конкретного пользователя. |
| Файл User\_Courses.cs | Данный файл содержит класс User\_Courses, который предназначен для хранения в себе информации о купленных курсах конкретного пользователя. |
| Файл User\_Test.cs | Данный файл содержит класс User\_Test, который предназначен для хранения в себе информации пройденных тестах конкретного пользователя. |

Подробная структура папки Styles представлена на рисунке 3.4.

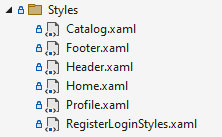


Рисунок 3.4 — Подробная структура папки «Styles»

Описание файлов представлено в таблице 3.4.

Таблица 3.4 – Описание файлов папки «Styles»

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл Catalog.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для каталогов приложения. |
| Файл Footer.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для подвала приложения. |

Окончание таблицы 3.4

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл Header.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для навигационного меню приложения. |
| Файл Home.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для главной страницы приложения. |
| Файл Profile.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для страницы профиля приложения. |
| Файл RegisterLoginStyles.xaml | Файл представляет собой словарь ресурсов, в котором хранятся стили для форм регистрации и авторизации приложения. |

Подробная структура папки View представлена на рисунке 3.5.

|  |
| --- |
|  |

Рисунок 3.5 — Подробная структура папки «View»

Описание файлов и внутренних папок представлено в таблице 3.5.

Таблица 3.5 – Описание файлов и внутренних папок папки «View»

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Папка admin | Папка хранит в себе окна и страницы, предназначенные для приложения администратора. |
| Файл Catalog.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех алгоритмов. |

Продолжение таблицы 3.5

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл Courses.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех курсов. |
| Файл FormBuy.xaml | Файл содержит в себе описание окна, в кором находится форма покупки. |
| Файл FormLoginRegister.xaml | Файл содержит в себе описание окна, в кором находится форма регистрации и авторизации. |
| Файл Home.xaml | Файл содержит в себе определение для страницы, в которой находится главная страница. |
| Файл Login.xaml | Файл содержит в себе определение для страницы, в которой реализована форма для авторизации. |
| Файл MainUser.xaml | Файл содержит в себе описание для главного окна, в котором содержится навигационное меню, подвал приложения. |
| Файл Profile.xaml | Файл содержит в себе определение для страницы, в которой реализован профиль. |
| Файл Register.xaml | Файл содержит в себе определение для страницы, в которой реализована форма регистрации. |
| Файл Test.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех тестов. |
| Файл AddAlgorithm.xaml | Файл содержит в себе описание страницы, в которой содержится форма добавления алгоритма. |
| Файл AddCourse.xaml | Файл содержит в себе описание страницы, в которой содержится форма добавления курса. |
| Файл AddTest.xaml | Файл содержит в себе описание страницы, в которой содержится форма добавления теста. |
| Файл AnswerAdmin.xaml | Файл содержит в себе описание страницы, в которой содержится форма ответа на вопрос пользователя. |
| Файл CatalogAdmin.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех алгоритмов приложения администратора. |
| Файл ChangeAlgorithm.xaml | Файл содержит в себе описание страницы, в которой содержится форма изменения алгоритма. |

Окончание таблицы 3.5

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл ChangeCourse.xaml | Файл содержит в себе описание страницы, в которой содержится форма изменения курса. |
| Файл ChangeTest.xaml | Файл содержит в себе описание страницы, в которой содержится форма изменения теста. |
| Файл CoursesAdmin.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех курсов приложения администратора. |
| Файл BasketPage.xaml | Файл содержит определение страницы для корзины книг. |
| Файл CatalogPage.xaml | Файл содержит определение страницы для каталога книг. |
| Файл LvlUpFormAdmin.xaml | Файл содержит в себе описание страницы, в которой содержится форма увеличения уровня пользователя. |
| Файл MainAdmin.xaml | Файл содержит в себе описание для главного окна, в котором содержится навигационное меню, подвал приложения администратора. |
| Файл QuestionsAdmin.xaml | Файл содержит определение окна, в котором описана форма для изменения номера телефона. |
| Файл TestAdmin.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех тестов приложения администратора. |
| Файл UsersAndTestsAdmin.xaml | Файл содержит в себе описание страницы, в которой идет перечисление всех тестов пройденных пользователями. |

В зависимости от того, под какой ролью зайдет пользователь, администратора или обычного пользователя, для него откроется определенное приложение, то есть пользовательское приложение или же приложение администратора.

Подробная структура папки ViewModel представлена на рисунке 3.6.

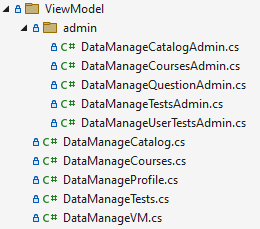


Рисунок 3.6 — Подробная структура папки «ViewModel»

Описание файлов и внутренних папок представлено в таблице 3.6.

Таблица 3.6 – Описание файлов и внутренних папок папки «ViewModel»

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Папка Admin | Папка хранит в себе классы, которые описывают взаимодействие пользователя с графической частью приложения администратора и бизнес логикой. |
| Файл DataManageCatalogAdmin.cs | Файл содержит класс, который описывает взаимодействие со страницей CatalogAdmin. |
| Файл DataManageCoursesAdmin.cs | Файл содержит класс, который описывает взаимодействие со страницей CoursesAdmin. |
| Файл DataManageQuestionAdmin.cs | Файл содержит класс, который описывает взаимодействие со страницей QuestionAdmin. |
| Файл DataManageTestsAdmin.cs | Файл содержит класс, который описывает взаимодействие со страницей TestsAdmin. |
| Файл DataManageUserTestsAdmin.cs | Файл содержит класс, который описывает взаимодействие со страницей UsersAndTestsAdmin. |
| Файл DataManageCatalog.cs | Файл содержит класс, который описывает взаимодействие со страницей Catalog. |
| Файл DataManageCourses.cs | Файл содержит класс, который описывает взаимодействие со страницей Courses. |

Окончание таблицы 3.6

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Файл DataManageProfile.cs | Файл содержит класс, который описывает взаимодействие с окном Profile. |
| Файл DataManageTests.cs | Файл содержит класс, который описывает взаимодействие со страницей Tests. |
| Файл DataManageVM.cs | Файл содержит класс, который описывает взаимодействие со страницей FormLoginRegister. |

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

## 3.2 Взаимоотношения между классами

Для визуализации взаимосвязей между классами используется диаграмма UML.

UML (Unified Modeling Language) диаграмма классов - это графическое представление структуры и отношений между классами в системе. Она позволяет визуализировать классы, их атрибуты, методы и связи между ними. UML диаграмма классов является мощным инструментом для анализа, проектирования и документирования структуры программных систем. Она помогает разработчикам и аналитикам лучше понять взаимодействие классов в системе и способствует созданию более эффективного и модульного кода.

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 А.

## 3.3 Модель базы данных

Для реализации поставленной задачи была создана база данных Algorithm\_Adventure. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных состоит из таблиц, представленных на рисунке 3.7. Все таблицы выполняют определенную роль в приложении, и соответствую требованиям, поставленным выше.

База данных разработана с учетом возможных изменений и расширений в будущем. Структура базы данных позволяет легко добавлять новые таблицы, поля или связи, а также вносить изменения в существующую структуру без необходимости значительных изменений в коде приложения.

В базе данных были определены ограничения целостности, которые обеспечивают правильность и согласованность данных. Это включает ограничения на значения полей, связи между таблицами и другие правила, которые гарантируют целостность данных в базе данных.

Скрипт для создания базы данных представлен в приложении В.

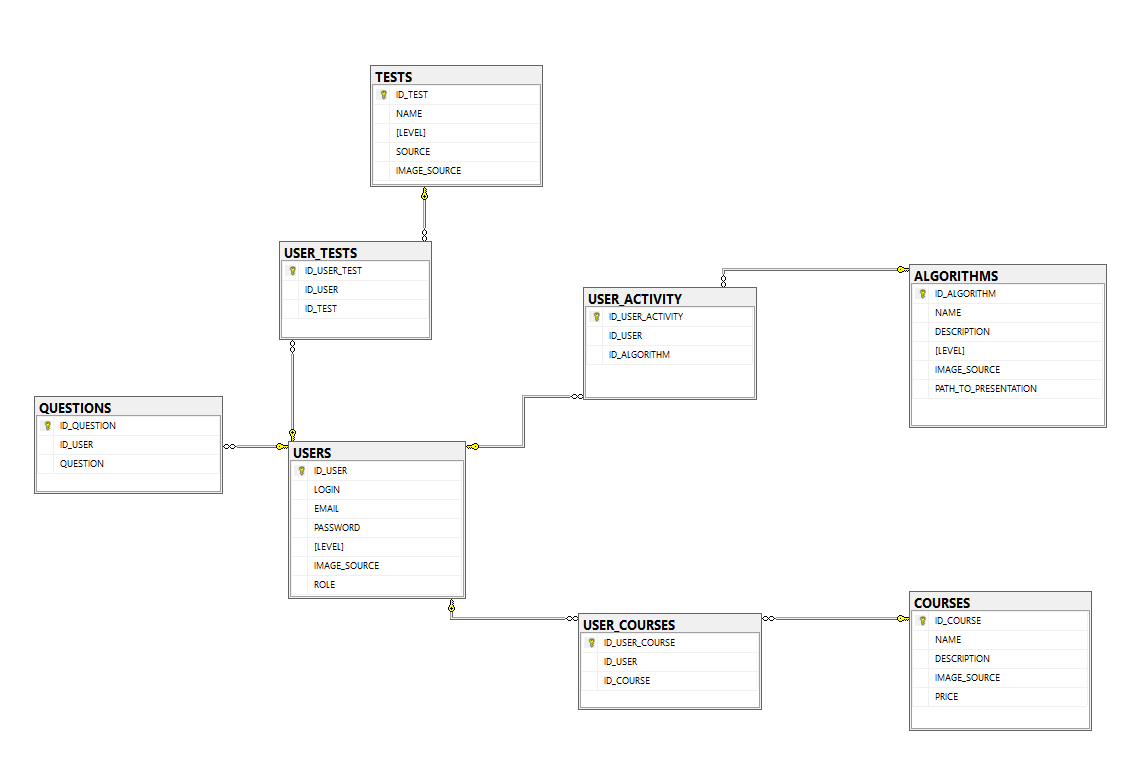


Рисунок 3.7 — База данных приложения

Таблица USERS изображена на рисунке 3.8. Данная таблица содержит в себе информацию о пользователе, которая представлена следующими столбцами: ID\_USER: уникальный номер пользователя, IMAGE\_PROFILE: путь к изображению пользователя, LOGIN: логин пользователя, PASSWORD: хранит пароль пользователя, EMAIL: электронная почта, ROLE: роль текущего пользователя (обычный или администратор), LEVEL: уровень пользователя.

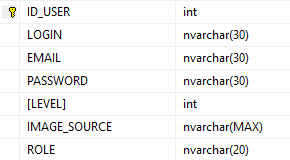


Рисунок 3.8 — Структура таблицы USERS

Таблица ALGORITHM изображена на рисунке 3.9. Данная таблица содержит в себе информацию об алгоритмах, которая представлена следующими столбцами: ID\_ALGORITHM: уникальный номер книги, IMAGE\_SOURCE: содержит путь к изображению, NAME: содержит название алгоритма, DESCRIPTION: описание, LEVEL: минимальный уровень для изучения, PATH\_TO\_PRESENTATION: путь к презентации.

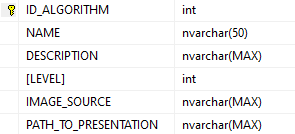


Рисунок 3.9 — Структура таблицы ALGORITHM

Таблица USER\_ACTIVITY изображена на рисунке 3.10. Данная таблица содержит в себе перечисление соотношения пройденных алгоритмов к пользователям и состоит из следующих столбцов: ID\_USER\_ACTIVITY: уникальный номер, ID\_USER: уникальный номер пользователя, ID\_ALGORITHM: уникальный номер алгоритма.

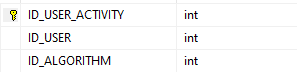


Рисунок 3.10 — Структура таблицы USER\_ACTIVITY

Таблица COURSES изображена на рисунке 3.11. Данная таблица содержит в себе информацию о курсах и состоит из следующих столбцов: ID\_COURSE: уникальный номер курса, IMAGE\_SOURCE: содержит путь к изображению, NAME: содержит название алгоритма, DESCRIPTION: описание, PRICE: цена.

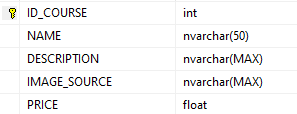


Рисунок 3.11 — Структура таблицы COURSES

Таблица USER\_COURSES изображена на рисунке 3.12. Данная таблица содержит в себе список курсов купленных пользователем и состоит из следующих столбцов: ID\_USER\_COURSES: уникальный номер, ID\_USER: уникальный номер пользователя, ID\_COURSE: уникальный номер курсов.



Рисунок 3.12 — Структура таблицы USER\_COURSES

Таблица QUESTIONS изображена на рисунке 3.13. Данная таблица содержит в себе информацию о вопросах и состоит из следующих столбцов: ID\_QUESTION: уникальный номер вопроса, ID\_USER: уникальный номер пользователя, QUESTION: вопрос.



Рисунок 3.13 — Структура таблицы QUESTIONS

Таблица TESTS изображена на рисунке 3.14. Данная таблица содержит в себе информацию о тестах и состоит из следующих столбцов: ID\_TEST: уникальный номер теста, NAME: название теста, LEVEL: минимальный допустимый уровень, SOURCE: ссылка на тест, IMAGE\_SOURCE: содержит путь к изображению.

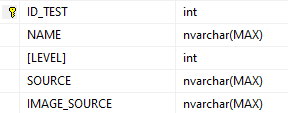


Рисунок 3.14 — Структура таблицы TESTS

Таблица USER\_TEST изображена на рисунке 3.15. Данная таблица представляет собой перечисление тестов написанных пользователями и состоит из столбцов: ID\_USER\_TEST: уникальный номер, ID\_USER: уникальный номер пользователя, ID\_TEST: уникальный номер теста.

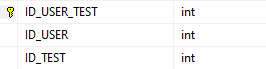


Рисунок 3.15 — Структура таблицы USER\_TEST

База данных была спроектирована с учетом принципов нормализации данных, что позволяет избежать избыточности и несогласованности данных. Каждая таблица имеет четко определенные поля и связи с другими таблицами для эффективного хранения и управления информацией.

Таким образом, структура базы данных «Algorithm\_Adventure» позволяет разработать приложение таким образом, чтобы все требование, установленные выше, были соблюдены. Данная база данных позволит хранить информацию о пользователях, курсах, тестах и прочем, что необходимо в данном приложении.

## 3.4 Проектирование архитектуры приложения

Для описания функциональности системы и того, как она используется различными группами пользователей, применяется диаграмма использования. Эта диаграмма представляет информацию о том, какие возможности и функции доступны каждой группе пользователей. В диаграмме использования используются два основных типа элементов: варианты использования и актёры.

Актёры представляют разные группы пользователей системы и могут быть любыми сущностями, которые взаимодействуют с системой. Варианты использования, с другой стороны, представляют собой функции или действия, которые пользователи могут выполнить в системе. Каждый вариант использования определяет набор шагов, которые актёры могут выполнить для взаимодействия с системой, и описывает, какие действия системы выполняются в ответ на эти действия пользователей.

Диаграмма использования является эффективным средством коммуникации между разработчиками, заказчиками и заинтересованными сторонами. Она помогает визуально представить функциональные требования системы и облегчает обсуждение и понимание функциональности между участниками проекта.

В целом, диаграмма использования является полезным инструментом для анализа требований, проектирования системы и коммуникации между участниками проекта. Она помогает лучше понять функциональность системы и ее взаимодействие с пользователями, что способствует успешной разработке программного продукта.

Диаграмма использования представлена в приложении Б.

## 3.5 Проектирование последовательностей проекта

Для наглядного представления взаимодействия между объектами системы в различные моменты времени, когда выполняется определенный сценарий использования, применяется диаграмма последовательностей в языке моделирования UML. Эта диаграмма позволяет наглядно показать, как объекты системы обмениваются сообщениями и взаимодействуют друг с другом для достижения определенной цели. Она отображает порядок и временную последовательность этих взаимодействий, позволяя лучше понять, как система функционирует в рамках конкретного сценария использования.

Для наглядного отображения временной последовательности взаимодействия объектов в системе используется диаграмма последовательностей в языке моделирования UML. На этой диаграмме каждый объект представлен вертикальной линией, называемой "линией жизни", которая показывает его существование в системе в определенный период времени. Объекты изображаются в виде прямоугольников, а сообщения между ними представлены стрелками или линиями.

Диаграмма последовательностей помогает наглядно представить взаимодействие между объектами и последовательность выполняемых ими действий во времени. Она является полезным инструментом для анализа и проектирования системы, а также служит средством коммуникации между разработчиками и заинтересованными сторонами, позволяя лучше понять, как система функционирует в различных сценариях использования.

Пример диаграммы последовательности представлен на рисунке 3.16.

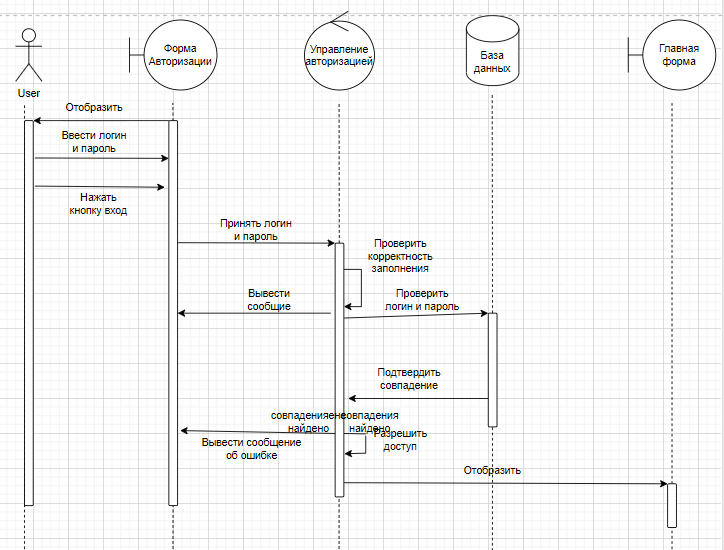


Рисунок 3.16 — Диаграмма последовательности

Пример диаграммы последовательности отображает процесс авторизации в приложении.

# 4 Реализация программного средства

## 4.1 Основные классы программного средства

Для выполнения технических задач программного средства для обучения алгоритмизации и программирования должны быть реализованы следующие функции:

* авторизация;
* регистрация;
* скачивание презентаций;
* возможность задать вопрос администратору;
* просмотр профиля;
* заполнение формы заказа курсов;
* редактирование профиля;
* прохождение теста;
* функционирование приложения администратора.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал, и реализация.

## 4.2 Описание классов и методов программного средства

### 4.2.1 Авторизация

Необходимый функционал для возможности входа пользователя в аккаунт, а также для проверки корректности данных, реализован в классе DataManageVM. После нажатия кнопки «Авторизоваться» сработает команда «LoginCommand».

В классе DataManageVM команда LoginCommand вызывает методы CheckUserLogin и CheckUserPassword, который отправляет запрос к базе данных на совпадение комбинации логина и пароля. Код метода CheckUserLogin изображен в листинге 4.1.

|  |
| --- |
| public static bool CheckUserLogin(string login)  {  using (ApplicationContext db = new ApplicationContext())  {  AppSettings.\_possibleLogin = login;  return db.USERS.FirstOrDefault(el => el.LOGIN == login) == null ? false : true;  }  } |

Листинг 4.1 — Метод CheckUserLogin

В случае успешного запроса к базе данных проверяется роль пользователя, то есть если пользователь имеет роль «пользователь», то для него открывается пользовательское приложение, иначе приложение администратора.

### 4.2.2 Регистрация

Для совершения пользователем регистрации также предназначен класс DataManageVM.

При нажатии на кнопку «Зарегистрироваться», сработает команда RegisterCommand. Данная команда вызывает метод CreateUser, если все поля прошли проверки. Код метода CreateUser отображен в листинге 4.2.

|  |
| --- |
| public static void CreateUser(string login, string email, string password)  {  using (ApplicationContext db = new ApplicationContext())  {  if (!db.USERS.Any(el => el.LOGIN == login))  {  User newUser = new User { LOGIN = login, EMAIL = email, PASSWORD = password, LEVEL = 0, IMAGE\_SOURCE = "C:\\Users\\USER\\Desktop\\labs\\Algorithm\\Img\\UserLogo.png", ROLE = "user" };  db.USERS.Add(newUser);  db.SaveChanges();  }  }  } |

Листинг 4.2 — Метод CreateUser

В данном методе описывается запрос к базе данных, где сначала проверяется существует ли пользователь под таким именем пользователя, а после, создается новый пользователь и сразу же добавляется в базу данных.

### 4.2.3 Просмотр каталога

После успешной авторизации пользователю открывается окно с главной страницей, где расположено навигационное меню, содержимое главной страницы и футер. Для перехода в каталог можно нажать кнопку «Каталог» в навигационном меню, либо же кнопку «Погнали» на главной странице.

Каталог имеет 3 уровня, начиная с 0 и заканчивая 2. На каждом уровне изображены алгоритмы, если уровень пользователя совпадает с уровнем алгоритма или больше его, он может скачать презентацию. Для скачивания нужно дважды нажать на элемент, после чего его обработает обработчик событий ListViewItem\_MouseDoubleClick, который вызовет команду ClickAlgorithmLevel0.

Команда ClickAlgorithmLevel0 скачивает презентацию выбранного алгоритма в папку, путь которой укажет пользователь в контекстном меню.

### 4.2.4 Покупка курса

Любой курс является платный и имеет кнопку купить. Графическая часть формы оплаты представлена на страницу FormBuy, где в одноименном классе проверяются данные на корректность, и в случае успеха в базе данных добавляется запись в таблицу USER\_COURSES, а также отправляется сообщение на электронную почту пользователя с уведомлением об успешной покупке и информации о тесте.

### 4.2.5 Просмотр профиля

При нажатии на иконку профиля в навигационном меню приложения, пользователю представится выбор между выходом из аккаунта и перехода в профиль. При нажатии на кнопку «Профиль» откроется страница Profile. На данной странице пользователь может изменить свою фотографию нажатием на кнопку «Изменить изображение», после чего вызовется команда ChangeImage, которое открывает меню диалога для выбора нового изображения и вызывает статический метод ChangeImage класса DataWorker, который применяет эти изменения для базы данных.

Также в профиле можно оставить вопрос администратору, узнать текущий уровень, количество скаченных алгоритмов и купленных курсов.

### 4.2.6 Добавление вопроса

Как упоминалось ранее, после перехода в профиль пользователь может оставить вопрос администратору, заполнив при этом соответствующее текстовое поле и нажав кнопку «Отправить». После этого вызовется команда AddQuestion, которая проверят не пустая ли строка и если условие корректно, то вызовет статический метод AddQuestion класса DataWorker, который внесет изменения в базу данных.

### 4.2.7 Приложение администратора

Для администраторов открывается собственное приложение, имеющее навигационное меню с различными кнопками. Приложение отображает список всех алгоритмов, тестов, курсов, вопросов и пользователей прошедших какие-либо тесты.

На странице CatalogAdmin выводится список всех алгоритмов. При двойном нажатии на алгоритм откроется форма для его изменения или удаления. Также на странице имеется кнопка добавить алгоритм. Схожий функционал реализован и на страницах CoursesAdmin и TestsAdmin.

На странице QuestionsAdmin выводится список всех вопросов и ID пользователей, от которых они поступили. При нажатии на кнопку «Ответить» рядом с вопросом, откроется форма с возможностью отправки ответа на почту пользователя.

На странице UsersAndTestsAdmin выводится список всех пользователей с их ID и ID тестов, которые они прошли. У каждого элемента списка есть кнопка «Посмотреть результаты», где можно перейти на тест и проверить как прошел его пользователь и администратор может принять решения исходя из результатов теста, повысить уровень пользователя или нет.

# 5 Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1 Тестирование авторизации и регистрации

В момент авторизации и регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, неверный пароль, незарегистрированный логин, некорректный адрес электронной почты. Также пользователь может вводить данные, которые уже были определены, например, логин, который есть уже у другого пользователя. Такие исключения обрабатываются программным средством с помощью показа пользователю всплывающих сообщений с текстом ошибки.

В случаях неправильного ввода пароля, логина или в случае пустых полей, в форме отображается всплывающее окно с соответствующим текстом. Пример тестирования в случае пустых полей или неправильной комбинации логина и пароля изображен на рисунке 5.1.

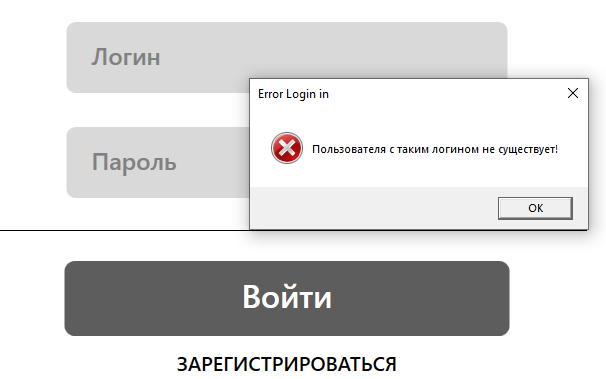


Рисунок 5.1 — Результат первого тестирования авторизации

При регистрации надо указывать следующие поля: логин, электронная почта, пароль. Каждое поле данной страницы обрабатывает собственные ошибки.

Результат тестирования при вводе логина длиной менее 6-ти символов приведен на рисунке 5.2.

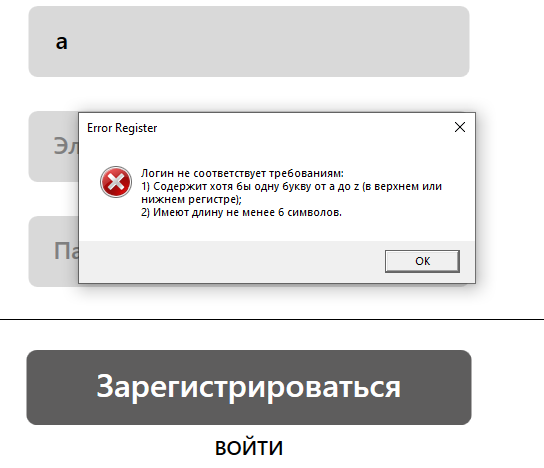


Рисунок 5.2 — Результат первого тестирования логина

Правило для формирования логина: можно использовать буквы алфавита, также можно использовать цифры, минимальная длина 6 символов, не может быть 2-ух повторяющихся логинов.

Результат тестирования при вводе в поле логина недопустимых символов приведен на рисунке 5.3.

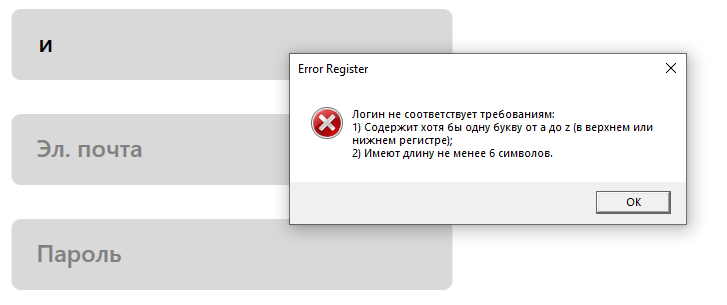


Рисунок 5.3 — Результат второго тестирования логина

Результат тестирования в случае пустого поля логина приведен на рисунке 5.4.

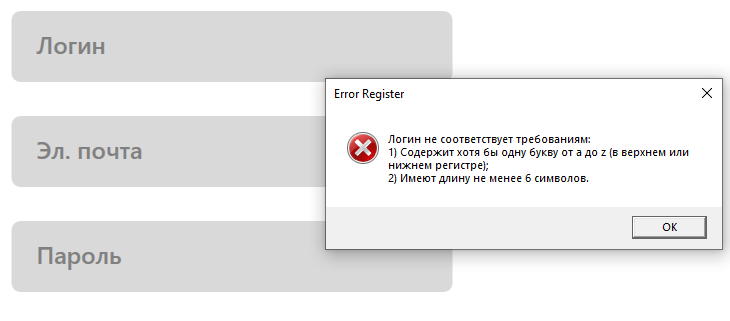


Рисунок 5.4 — Результат третьего тестирования логина

Правило для записи электронной почты: имя почты может содержать буквы английского алфавита, цифры и различные символы, обязательно должен быть символ «@» после которого идет корректный домен почты, также не может быть повторяющихся почт.

Результат тестирования при вводе почты без символа «@» приведен на рисунке 5.5.

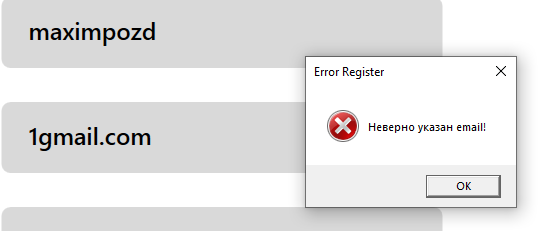


Рисунок 5.5 — Результат первого тестирования почты

Дополнительно в проверке валидности электронной почты идет проверка на существование данного домена.

Результат тестирования при вводе почты с недопустимыми символами приведен на рисунке 5.6.

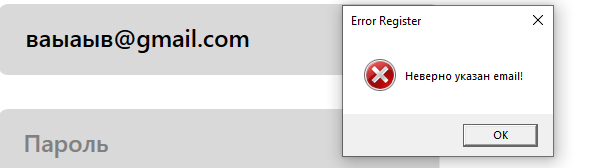


Рисунок 5.6 — Результат второго тестирования почты

Результат тестирования при пустом поле почты приведен на рисунке 5.7.

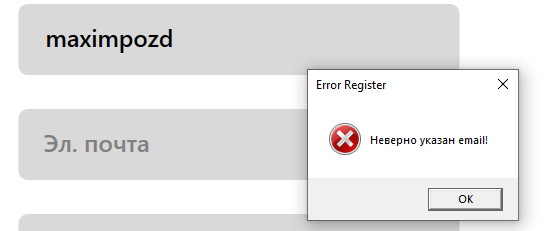


Рисунок 5.7 — Результат третьего тестирования почты

Правила для заполнения пароля: пароль может содержать различные символы, но обязательна прописная и строчная буква, также обязательно наличие цифр, минимальная длина пароля 6 символов.

Результат тестирования при вводе маленькой длины пароля приведен на рисунке 5.8.

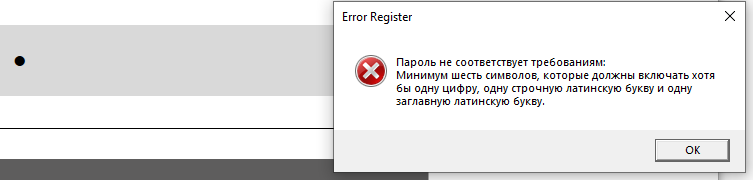


Рисунок 5.8 — Результат первого тестирования пароля

Пользователь должен сам придумать пароль как можно надежнее, опираясь на правила составления пароля в приложении. Но также не стоит придумывать пароль слишком большой длины.

Результат тестирования при вводе пароля, который не соответствует правилам, приведен на рисунке 5.9.

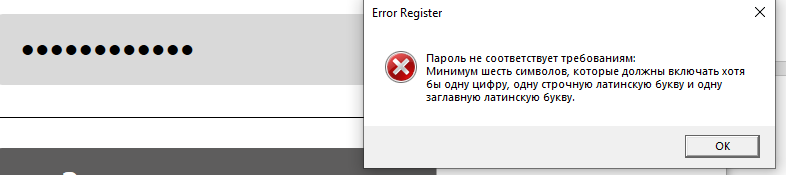


Рисунок 5.9 — Результат второго тестирования пароля

Таким образом было проведено успешное тестирование форм авторизации и регистрации.

## 5.2 Тестирование оплаты книги

При оплате книги пользователь должен вводить следующие поля: номер карты, срок действия карты и имя владельца.

Правила при заполнении номера карты: в приложении принимаются номера карт следующих типов: Visa, MasterCard, Maestro, Mir.

Результат тестирования при неправильном вводе карты приведен на рисунке 5.10.

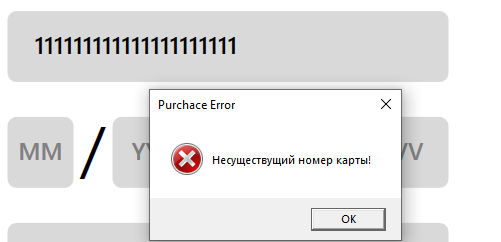


Рисунок 5.10 — Результат тестирования номера карты

Правила при заполнении даты: формат даты должен соответствовать формату MM/DD, где первая часть должны быть в промежутке от 1 до 12, а вторая от 23 до 29.

Результат тестирования при неправильном вводе даты приведен на рисунке 5.11.

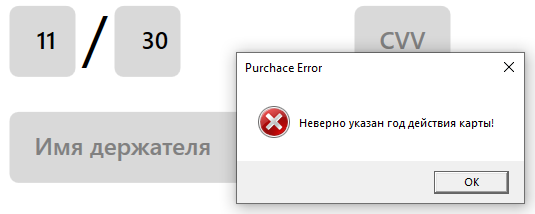


Рисунок 5.11 — Результат тестирования даты

Правила при заполнении имени владельца: допустимы только буквы английского алфавита.

Результат тестирования при неправильном вводе имени владельца приведен на рисунке 5.12.

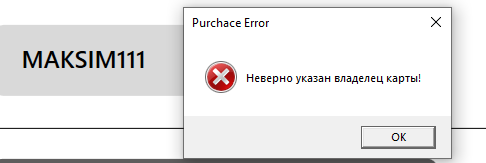


Рисунок 5.12 — Результат тестирования имени владельца

Также стоит рассмотреть различные случаи возникновения ошибок приложения, таких как повторной покупки курсов.

Результат тестирования при повторной покупки курсов приведен на рисунке 5.13.

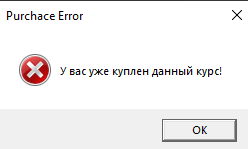


Рисунок 5.13 — Результат тестирования повторной покупки курсов

Также нельзя задавать пустой вопрос администратору.

Результат тестирования пустого поля вопроса приведен на рисунке 5.14.

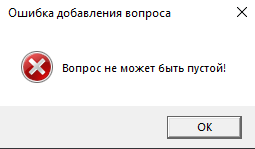


Рисунок 5.14 — Результат тестирования пустого поля вопроса

Таким образом было проведено успешное тестирование пользовательского приложения.

## 5.3 Тестирование приложения администратора

В приложении администратора также возможны исключительные ситуации, которые нужно отловить и сообщить об ошибке. Результаты тестирования таких ситуаций представлены ниже.

Результат тестирования при попытке отправления пустого ответа приведен на рисунке 5.15.

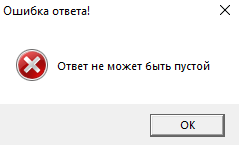


Рисунок 5.15 — Результат тестирования отправки пустого ответа

Результат тестирования при попытке добавление курса с пустым названием приведен на рисунке 5.16.

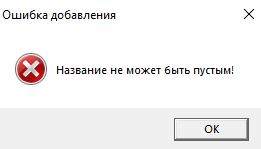


Рисунок 5.16 — Результат первого тестирования добавления курса

Результат тестирования при попытке добавления курса без цены на рисунке 5.17.

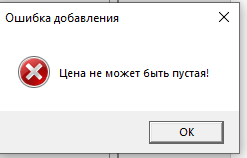


Рисунок 5.17 — Результат второго тестирования добавления курса

Все последующие возможные ошибки в приложении были успешно протестированы. По результатам тестирования, можно сказать, что приложения пользователя и администратора работают корректно.

# 6 Руководство по установке и использованию

При первом запуске приложения будет открыто окно авторизации, которое показано на рисунке 6.1.

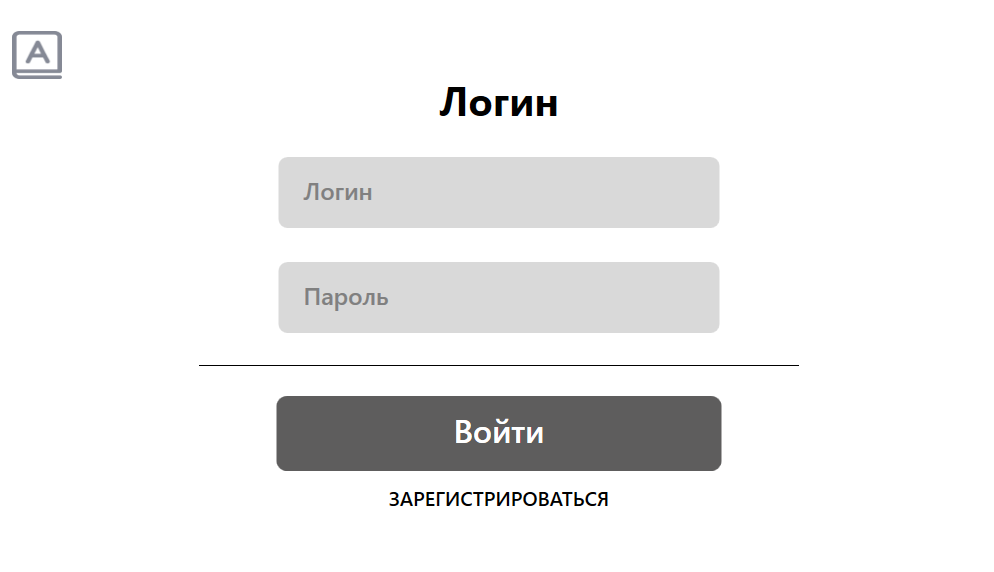


Рисунок 6.1 — Окно авторизации

В том случае, если у пользователя нет зарегистрированного аккаунта, он может нажать на кнопку «ЗАРЕГИСТРИРОВАТЬСЯ», и пройти там регистрацию.

После нажатия на кнопку «ЗАРЕГИСТРИРОВАТЬСЯ» откроется страница для регистрации, в котором все поля должны быть заполнены в соответствии с правилами: логин: буквы английского алфавита, а также цифры, электронная почта: корректный адрес электронной почты, пароль: разные символы, но обязательно наличие прописных и строчных букв, а также цифр. Окно регистрации изображено на рисунке 6.2.



Рисунок 6.2 — Окно регистрации

После регистрации пользователь должен войти с свой аккаунт, нажав на кнопку «ВОЙТИ». Нового пользователя сразу встречает главная страница пользовательского приложения, с бодрым текстом и кнопкой «Погнали!». Сверху располагается навигационное меню, которое имеет кнопки перемещения по страницам.

Главная страница представлена на рисунке 6.3.

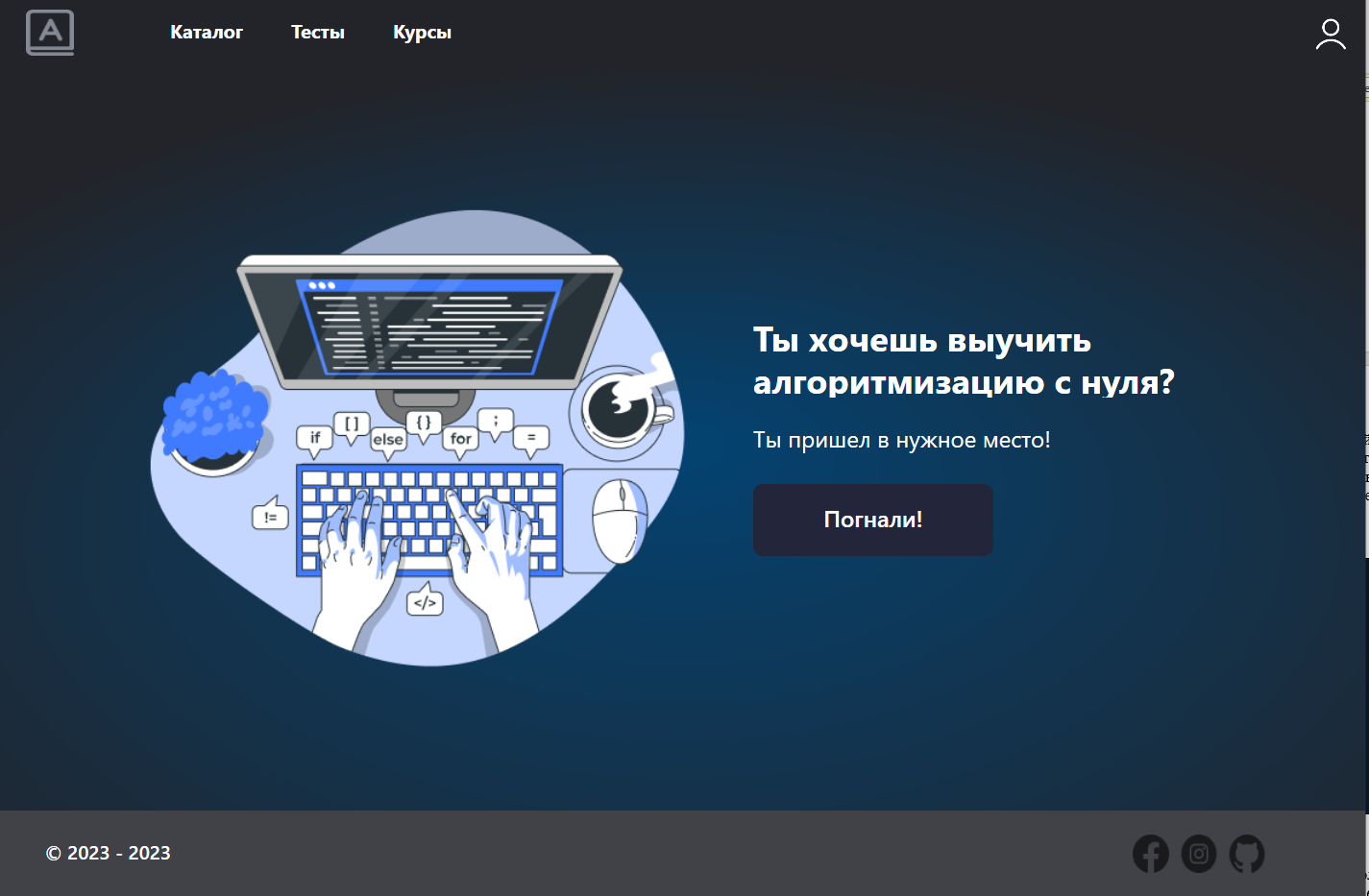


Рисунок 6.3 — Главная страница

Следующая кнопка справа от главной страницы переведет пользователя на новую страницу каталога алгоритмов. Каталог алгоритмов представлен на рисунке 6.4.

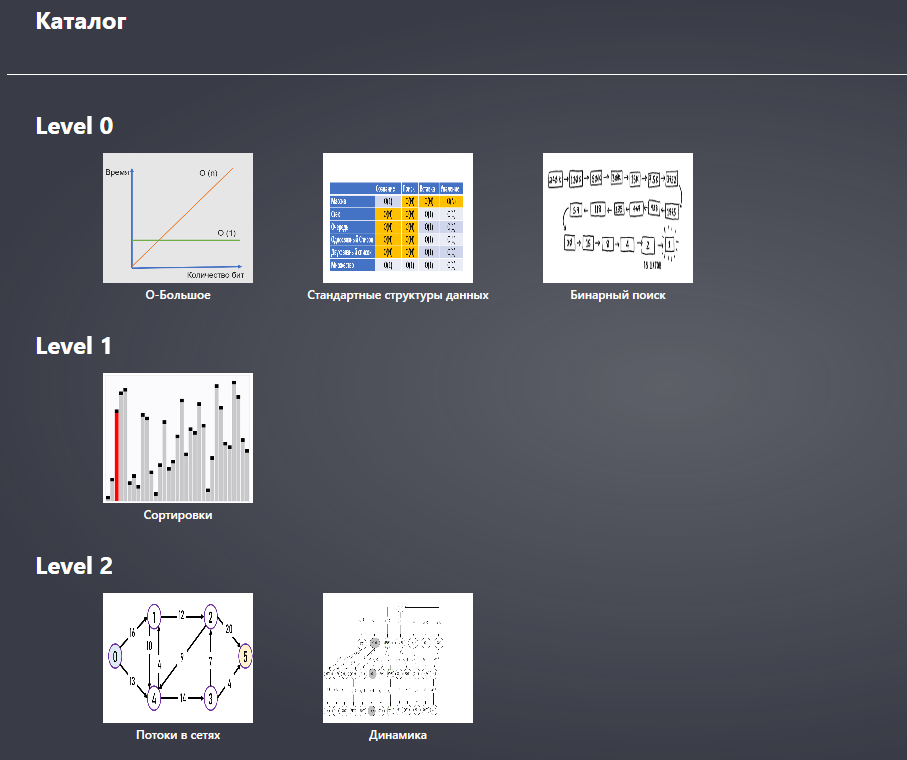


Рисунок 6.4 — Каталог алгоритмов

Следующая кнопка — это каталог тестов, перейдя в который пользователь может выбрать тест двойным нажатием и его перенаправит на сайт, где он сможет его пройти.

Каталог тестов изображен на рисунке 6.5.

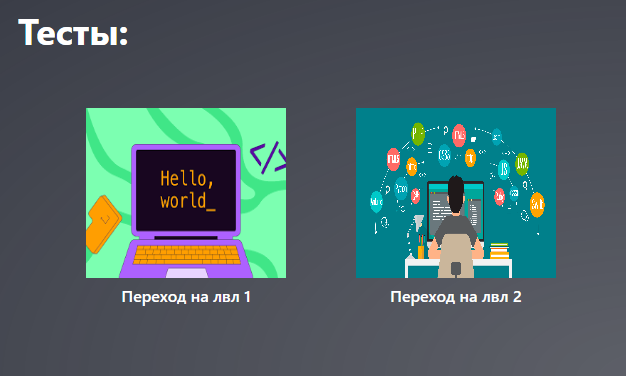


Рисунок 6.5 — Каталог тестов

Следующая кнопка — это каталог курсов, перейдя в который пользователь может купить курс нажатием на кнопку, где у него откроется форма для покупки. Каталог тестов изображен на рисунке 6.6.

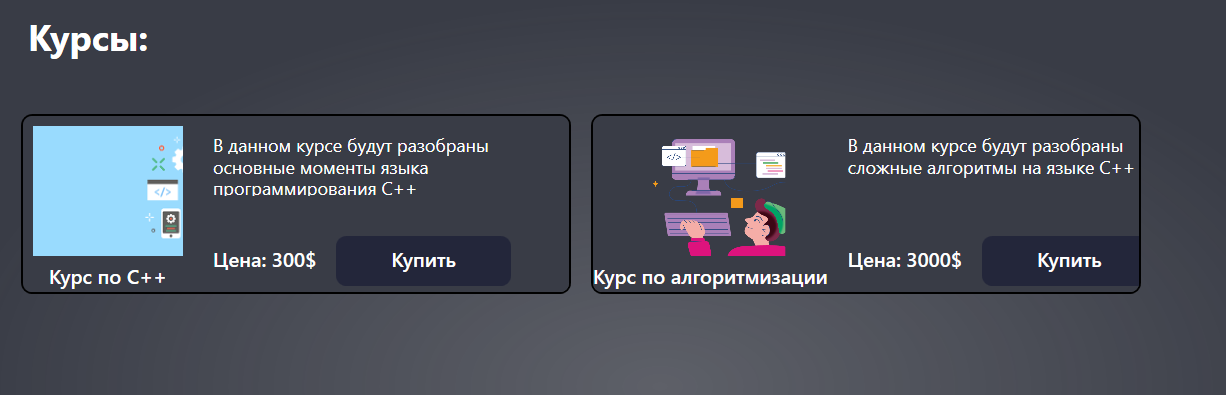


Рисунок 6.6 — Каталог тестов

Форма покупки имеет следующие поля: номер карточки, ММ (месяц), YY (год), CVV (трехзначный код), имя держателя.

Форма для покупки представлена на рисунке 6.7.

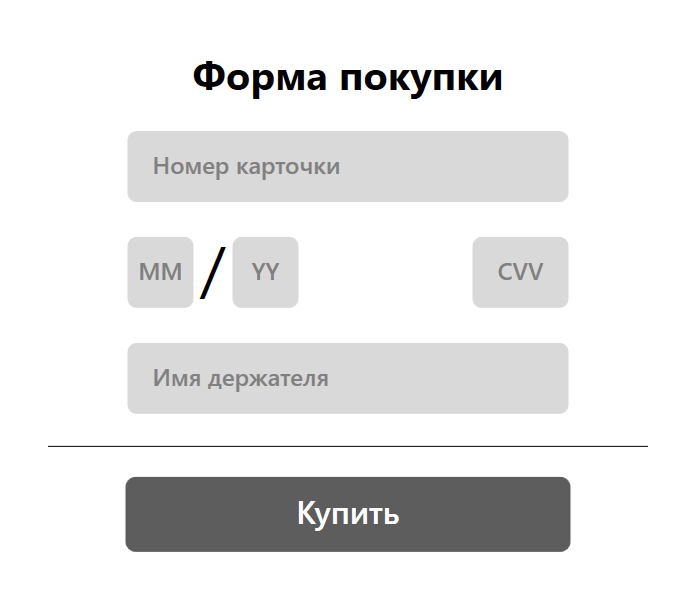


Рисунок 6.7 — Форма для покупки

Следующая кнопка — это кнопка профиля, при нажатии на которую появятся 2 кнопки, одна перенаправит на страницу профиля, другая выйдет из аккаунта. Страница профиля изображена на рисунке 6.8.

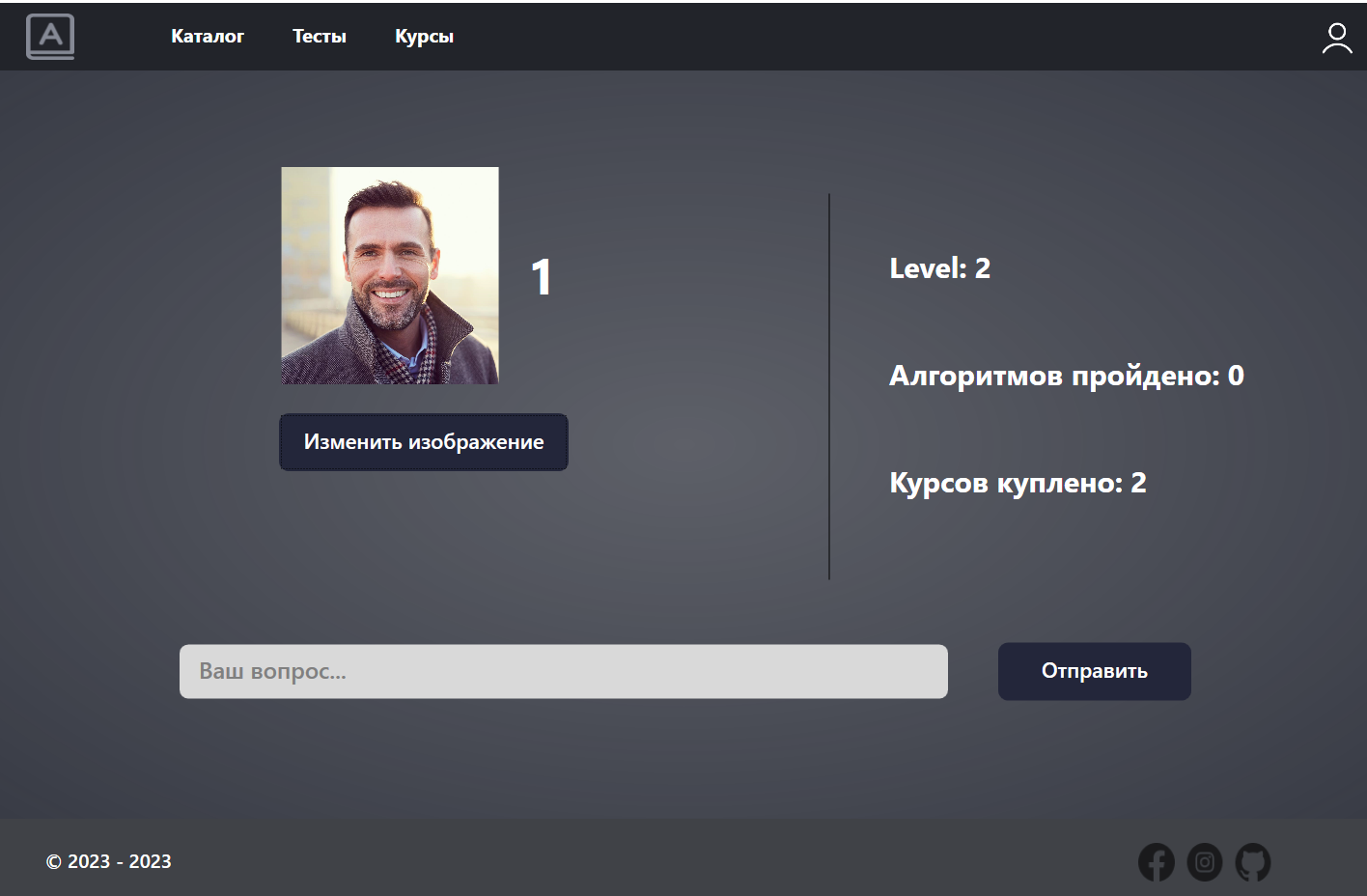


Рисунок 6.8 — Страница профиля

Важно помнить, для того, чтобы пользоваться данным приложением нужно иметь доступ в Интернет.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было успешно разработано программное средство для обучения алгоритмизации и программирования. Это приложение предоставляет удобный и многофункциональный интерфейс для изучения материала, а также позволяет администраторам управлять контентом приложения.

В ходе разработки были рассмотрены и применены современные технологии, такие как WPF, Entity Framework Core и MS SQL, что позволило создать надежное и масштабируемое программное решение. Также применение архитектурного паттерна MVVM обеспечило разделение логики и пользовательского интерфейса, что повысило гибкость и переиспользуемость кода.

Программное средство предоставляет широкий набор функциональных возможностей для пользователей, включая регистрацию, авторизацию, скачивание алгоритмов, покупку курсов, возможность задать вопрос администратору.

Было разработано приложение для администратора, которое включает в себя такие функции, как просмотр каталога алгоритмов, тестов, курсов, их добавление, изменение и удаление, ответы на вопросы пользователей путем отправки сообщения на электронную почту.

Было проведено тестирование всех возможных функций. В ходе тестирования были исправлены некоторые моменты, что сделало программное средство для обучения алгоритмизации и программированию еще надежнее для пользователя.

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

# Список использованных источников

1. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.04.2023
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. – Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 23.04.2023
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Режим доступа: Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 25.04.2023
5. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 25.04.2023
6. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.
7. Работа с Entity Framework Core [Электронный ресурс] – Режим доступа: <https://professorweb.ru/my/entity-framework/6/level1/> − Дата доступа 28.04.2023

# ПРИЛОЖЕНИЕ А

Диаграмма классов

|  |
| --- |
|  |

# ПРИЛОЖЕНИЕ Б

Диаграмма последовательности

|  |
| --- |
|  |

# ПРИЛОЖЕНИЕ В

Листинг 1: Создание базы данных

|  |
| --- |
| create database Algorithm\_Adventure |

Листинг 2: Создание таблицы COURSES

|  |
| --- |
| create table COURSES  (  ID\_COURCE int primary key identity(1, 1),  NAME nvarchar(50),  DESCRIPTION nvarchar(max),  IMAGE\_SOURCE nvarchar(max),  PRICE float  ) |

Листинг 3: Создание таблицы USERS

|  |
| --- |
| create table USERS  (  ID\_USER int primary key identity(1, 1),  LOGIN nvarchar(30),  EMAIL nvarchar(30),  PASSWORD nvarchar(30),  LEVEL int,  IMAGE\_SOURCE nvarchar(max),  ROLE nvarchar(20)  ) |

Листинг 4: Создание таблицы ALGORITHMS

|  |
| --- |
| create table ALGORITHMS  (  ID\_ALGORITHM int primary key identity(1, 1),  NAME nvarchar(50),  DESCRIPTION nvarchar(max),  LEVEL int,  IMAGE\_SOURCE nvarchar(max),  PATH\_TO\_PRESENTATION nvarchar(max)  ) |

Листинг 5: Создание таблицы QUESTIONS

|  |
| --- |
| create table QUESTIONS  (  ID\_QUESTION int primary key identity(1, 1),  ID\_USER int foreign key references USERS(ID\_USER),  QUESTION nvarchar(max),  ) |

Листинг 6: Создание таблицы TESTS

|  |
| --- |
| create table TESTS  (  ID\_TEST int primary key identity(1, 1),  NAME nvarchar(max),  LEVEL int,  SOURCE nvarchar(max),  IMAGE\_PROFILE nvarchar(max)  ) |

Листинг 7: Создание таблицы USER\_ACTIVITY

|  |
| --- |
| create table USER\_ACTIVITY  (  ID\_USER\_ACTIVITY int primary key identity(1, 1),  ID\_USER int foreign key references USERS(ID\_USER),  ID\_ALGORITHM int foreign key references ALGORITHMS(ID\_ALGORITHM)  ) |

Листинг 8: Создание таблицы USER\_COURSES

|  |
| --- |
| create table USER\_COURSES  (  ID\_USER\_COURSE int primary key identity(1, 1),  ID\_USER int foreign key references USERS(ID\_USER),  ID\_COURSE int foreign key references COURSES(ID\_COURSE)  ) |

Листинг 9: Создание таблицы USER\_TESTS

|  |
| --- |
| create table USER\_TESTS  (  ID\_USER\_TESTS int primary key identity(1, 1),  ID\_USER int foreign key references COURSES(ID\_COURSE),  ID\_TEST int foreign key references TESTS(ID\_TEST)  ) |

Листинг 10: Форма авторизации

|  |
| --- |
| <Grid>    <StackPanel x:Name="LoginPanel">  <Image Source="C:\Users\USER\Desktop\labs\Algorithm\Img\Logo.png"  Margin="25 25 0 0"  Style="{DynamicResource Logo}"/>  <TextBlock Text="Авторизация"  Style="{DynamicResource TextBlockRegister}"/>  <Border Margin="0, 30, 0, 0"  Style="{DynamicResource BorderForTextBox}">  <Grid>  <TextBox x:Name="LoginForLogin"  Style="{DynamicResource TextBoxRegister}"  TextChanged="LoginForLogin\_TextChanged"/>  <TextBlock x:Name="LoginForLoginPlug"  Text="Логин"  Style="{DynamicResource TextBlockPlug}"/>  </Grid>  </Border>  <Border Margin="0, 30, 0, 0"  Style="{DynamicResource BorderForTextBox}">  <Grid>  <PasswordBox x:Name="PasswordForLogin"  Style="{DynamicResource PasswordBoxRegister}"  PasswordChanged="PasswordForLogin\_PasswordChanged" />  <TextBlock x:Name="PasswordForLoginPlug"  Text="Пароль"  Style="{DynamicResource TextBlockPlug}"/>  </Grid>  </Border>  <Border Margin="0 30 0 0"  Style="{DynamicResource Line}"/>  <Button Content="Войти"  Margin="0 30 0 0"  Style="{DynamicResource ButtonRegister}"  Template="{DynamicResource btTemplate}"  Command="{Binding DataContext.LoginCommand, RelativeSource={RelativeSource AncestorType=Page}}"  CommandParameter="{Binding ElementName=LoginPage}"/>    </StackPanel>    </Grid> |

Листинг 11: Форма регистрации

|  |
| --- |
| <Grid>  <StackPanel x:Name="RegisterPanel">  <Image Source="C:\Users\USER\Desktop\labs\Algorithm\Img\Logo.png"  Margin="25 25 0 0"  Style="{DynamicResource Logo}"/>  <TextBlock Text="Регистрация"  Style="{DynamicResource TextBlockRegister}"/>  <Border Margin="0, 30, 0, 0"  Style="{DynamicResource BorderForTextBox}">  <Grid>  <TextBox x:Name="LoginForRegister"  Style="{DynamicResource TextBoxRegister}"  TextChanged="LoginForRegister\_TextChanged"/>  <TextBlock x:Name="LoginForRegisterPlug"  Text="Логин"  Style="{DynamicResource TextBlockPlug}"/>  </Grid>  </Border>  <Border Margin="0, 30, 0, 0"  Style="{DynamicResource BorderForTextBox}">  <Grid>  <TextBox x:Name="EmailForRegister"  Style="{DynamicResource TextBoxRegister}"  TextChanged="EmailForRegister\_TextChanged"/>  <TextBlock x:Name="EmailForRegisterPlug"  Text="Эл. почта"  Style="{DynamicResource TextBlockPlug}"/>  </Grid>  </Border>  <Border Margin="0, 30, 0, 0"  Style="{DynamicResource BorderForTextBox}">  <Grid>  <PasswordBox x:Name="PasswordForRegister"  Style="{DynamicResource PasswordBoxRegister}"  PasswordChanged="PasswordForRegister\_PasswordChanged"/>  <TextBlock x:Name="PasswordForRegisterPlug"  Text="Пароль"  Style="{DynamicResource TextBlockPlug}"/>  </Grid>  </Border>  <Border Margin="0 30 0 0"  Style="{DynamicResource Line}"/>  <Button Content="Зарегистрироваться"  Margin="0 30 0 0"  Style="{DynamicResource ButtonRegister}"  Template="{DynamicResource btTemplate}"  Command="{Binding DataContext.RegisterCommand, RelativeSource={RelativeSource AncestorType=Page}}"  CommandParameter="{Binding ElementName=RegisterPage}"/>    </StackPanel>  </Grid> |

Листинг 12: класс ApplicationContext

|  |
| --- |
| public class ApplicationContext: DbContext  {  public DbSet<User> USERS { get; set; }  public DbSet<Algorithm> ALGORITHMS { get; set; }  public DbSet<User\_Activity> USER\_ACTIVITY { get; set; }  public DbSet<Courses> COURSES { get; set; }  public DbSet<User\_Courses> USER\_COURSES { get; set; }  public DbSet<Question> QUESTIONS { get; set; }  public DbSet<Test> TESTS { get; set; }  public DbSet<User\_Test> USER\_TESTS { get; set; }  public ApplicationContext()  {  }  public ApplicationContext(DbContextOptions<ApplicationContext> options) : base(options)  {  }  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  optionsBuilder.UseSqlServer("Server=DESKTOP-P7BTVC4;Database=Algorithm\_Adventure;Trusted\_Connection=True;TrustServerCertificate=True;");  }  } |

Листинг 13: Методов, отсылающего сообщение на почту

public static MailMessage CreateMail(string name, string emailFrom, string emailTo, string subject, string body)

{

var from = new MailAddress(emailFrom, name);

var to = new MailAddress(emailTo);

var mail = new MailMessage(from, to);

mail.Subject = subject;

mail.Body = body;

mail.IsBodyHtml = true;

return mail;

}

public static void SendMail(string host, int smptPort, string emailFrom, string pass, MailMessage mail)

{

SmtpClient smtp = new SmtpClient(host, smptPort);

smtp.Credentials = new NetworkCredential(emailFrom, pass);

smtp.EnableSsl = true;

smtp.Send(mail);

}