## Аннотация

## Программа "Абитуриент" предназначена для автоматизации процесса приема заявок от абитуриентов на различные факультеты учебного заведения. Система обеспечивает абитуриентам возможность ввода своих баллов за экзамены, выбора предпочтительных факультетов и подачи заявок. Администраторы могут рассматривать заявки, принимать решения о зачислении или отклонении, а также управлять статусами заявок.

Программа предоставляет удобный интерфейс для взаимодействия абитуриентов и администраторов, обеспечивая эффективное взаимодействие и оперативное решение вопросов по приему на обучение. Важным элементом является также система уведомлений, которая сообщает абитуриентам о результатах рассмотрения их заявок.

Этот программный продукт спроектирован с учетом требований к надежности, обеспечивает контроль за вводимыми и выводимыми данными, а также поддерживает устойчивое функционирование в различных условиях эксплуатации.

## 

**Содержание**

## Введение……………………………………………………………………………...4

1. Назначение и область применения…………………………………………..….5
2. Постановка задачи…………………………………………………………..…....6
3. Описание программы………………………………………………………….....7
   1. Общие сведения……………………………………………………...........7
   2. Функциональное назначение………………………………..………..…..8
   3. Описание логической структуры………………………………….....…..8
   4. Используемые технические средства…………………………….……...9
   5. Вызов и загрузка………………………………………………………..…9
   6. Входные и выходные данные……………………………………..….…..9
4. Программа и методика испытаний…………………………………………..…10
   1. Объекты испытаний……………………………………………..…….…10
   2. Цель испытаний……………………………………………………...…..10
   3. Требования к программе…………………………………………...……10
   4. Методы испытаний…………………………………………………........10
5. Руководство оператора………………………………………………………...11
   1. Выполнение программы…………………………………………….…11
   2. Сообщение оператору……………………………………………….....11
6. Мероприятия по информационной безопасности ……..………………..…..12

Заключение……………………………………………………………………...…13

Приложения. Текст программы ...………………………………………………..14

Источники, использованные при разработке…………………………………....19

## 

**Введение**

## 

## Наименование программы: "Абитуриент".

Условное обозначение темы разработки: "Система приема заявок абитуриентов на факультеты учебного заведения".

Основание для разработки: Задание на разработку программного продукта от СПб ГБПОУ "Политехнический колледж городского хозяйства", утверждено 26.07.2023.

**1. Назначение и область применения**

## Программа "Абитуриент" предназначена для автоматизации процесса приема заявок от абитуриентов на учебные программы учебного заведения. Программа обеспечивает удобный интерфейс для абитуриентов, позволяя им вводить баллы за экзамены, выбирать факультеты и подавать заявки. Администраторы могут рассматривать заявки, принимать решения о зачислении и вести учет студентов, зачисленных на факультеты.

Характеристики области применения:

* Автоматизация приема заявок от абитуриентов.
* Учет статуса заявок и зачисленных студентов.
* Возможность ввода баллов за экзамены и выбора факультетов.
* Удобный интерфейс для абитуриентов и администраторов.

Существующие аналоги программы обычно представляют собой системы управления абитуриентскими данными в образовательных учреждениях. Однако, программа "Абитуриент" разрабатывается с учетом конкретных потребностей учебного заведения, обеспечивая персонализированный и гибкий подход к приему абитуриентов и учету заявок.

**2. Постановка задачи**

## Необходимо разработать программу "Абитуриент" для автоматизации процесса приема заявок от абитуриентов и учета студентов, зачисленных на факультеты. Программа должна обладать следующими функциональными характеристиками:

* Регистрация и аутентификация пользователей: абитуриенты могут регистрироваться в системе, предоставляя свои персональные данные, и идентифицироваться для доступа к функционалу.
* Ввод баллов за экзамены: абитуриенты должны иметь возможность вводить свои баллы за экзамены для участия в конкурсе на зачисление на факультет.
* Выбор факультета: абитуриенты могут выбирать факультет, на который они хотели бы поступить.
* Подача заявки: абитуриенты могут подавать заявки на участие в конкурсе на зачисление на выбранный факультет.
* Рассмотрение заявок администраторами: администраторы системы могут просматривать и рассматривать заявки абитуриентов, принимать решение о зачислении или отклонении.
* Учет зачисленных студентов: система должна вести учет студентов, зачисленных на факультеты, с указанием их персональных данных и статуса зачисления.
* Уведомления: в случае зачисления абитуриента на факультет, программа должна предоставлять уведомление абитуриенту и администратору системы.
* Математические методы в данной программе применяться не будут, поскольку основной упор делается на функциональность, связанную с обработкой данных и взаимодействием с пользователями.

Допущения и ограничения:

* Программа предназначена для использования в рамках конкретного учебного заведения и адаптирована под его потребности.
* Абитуриенты должны предоставлять достоверные персональные данные при регистрации.
* Для аутентификации могут использоваться логин и пароль.
* Администраторы системы имеют полномочия для просмотра и управления заявками абитуриентов.

Цель программы - обеспечить прозрачный и эффективный процесс приема абитуриентов и учета зачисленных студентов на факультеты учебного заведения.

## 

**3. Описание программы**

**3.1 Общие сведения**

Наименование программы: "Абитуриент"

Программное обеспечение:

Операционная система Windows (7 и выше);

Система управления базами данных Microsoft SQL Server;

Язык программирования C#.

Окна программы на рисунках (1-3).

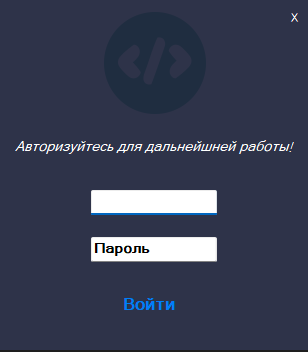


Рис.1 Окно авторизации.

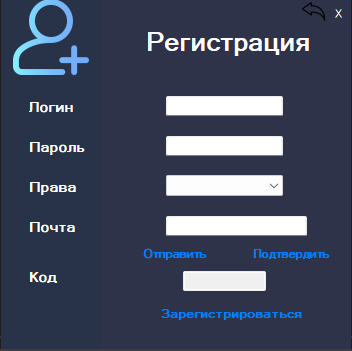


Рис.2 Окно регистрации

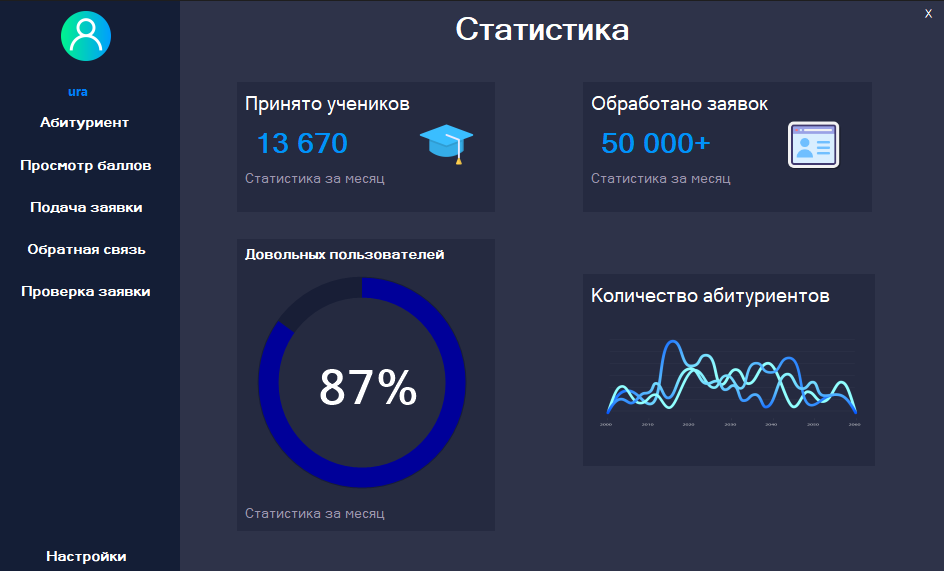
****

Рис.3 Окно пользователя

**3.2 Функциональное назначение**

Программа "Абитуриент" предназначена для автоматизации процесса приема заявок от абитуриентов и учета студентов, зачисленных на факультеты. Основные классы задач включают в себя:

* Регистрация и аутентификация пользователей;
* Ввод баллов за экзамены и выбор факультета;
* Подача заявки на участие в конкурсе;
* Рассмотрение заявок администраторами;
* Учет зачисленных студентов.

**3.3 Описание логической структуры**

Алгоритм программы основан на следующих принципах:

* Пользовательская регистрация и аутентификация через логин и пароль;
* Ввод баллов за экзамены и выбор факультета абитуриентами;
* Обработка заявок администраторами, принятие решения о зачислении или отклонении;
* Учет зачисленных студентов с указанием персональных данных и статуса зачисления.

**3.4 Используемые технические средства**

Программа "Абитуриент" разработана для использования на персональных компьютерах под управлением операционной системы Windows. В качестве системы управления базами данных используется Microsoft SQL Server.

**3.5 Вызов и загрузка**

Программа вызывается из рабочего стола или меню пуск. Загрузка осуществляется при запуске программы из исполняемого файла. Входные точки в программу предоставляются через пользовательский интерфейс, предоставляющий функционал по вводу данных абитуриентами и администраторами.

**3.6 Входные и выходные данные**

Входные данные представляют собой персональные данные абитуриентов, баллы за экзамены и выбор факультета. Выходные данные включают в себя уведомления о результатах рассмотрения заявок и статусе зачисления. Формат данных предусматривает использование текстовых полей, числовых значений и булевых флагов для обеспечения удобства ввода и визуализации информации.

**4. Программа и методика испытаний**

**4.1 Объекты испытаний**

**Наименование**: Программа "Абитуриент"

**Область применения**: Программа предназначена для использования абитуриентами, администраторами и другими участниками процесса приема на факультет.

Обозначение испытуемой программы: AbiturientProgram

**4.2 Цель испытаний**

## **Цель**: Проверка корректности работы основных функций программы "Абитуриент", таких как подача заявки, рассмотрение заявок администратором и уведомление абитуриентов о результатах зачисления.

**4.3 Требования к программе**

Требования:

* Программа должна корректно обрабатывать ввод баллов абитуриентами.
* Администратор должен иметь возможность просматривать и принимать/отклонять заявки.
* Абитуриенты должны получать уведомления о результатах зачисления.
* Программа должна обеспечивать безопасность хранения данных.

**4.4 Методы испытаний**

**Методы испытаний**:

* Тестирование ввода данных: Проверка корректности обработки ввода баллов абитуриентами.
* Тестирование административных функций: Проверка возможности администратора просматривать и обрабатывать заявки.
* Тестирование уведомлений: Проверка корректности уведомлений, отправляемых абитуриентам о результатах зачисления.
* Тестирование безопасности: Проверка уровня безопасности программы, включая защиту данных пользователей.

**5. Руководство оператора**

**5.1 Выполнение программы**

Последовательность действий оператора:

* Запустите программу "Абитуриент" на компьютере оператора.
* Введите учетные данные для входа в систему (если требуется).
* Выберите необходимые функции из предоставленного интерфейса программы (подача заявки, рассмотрение заявок и т.д.).
* Вводите данные согласно предоставленным инструкциям и подтверждайте действия.

Функции, команды и возможные варианты:

* Подача заявки: Абитуриент вводит свои баллы и подает заявку на факультет.
* Рассмотрение заявок: Администратор просматривает заявки, решает принять или отклонить.
* Уведомления: Абитуриенты получают уведомления о результатах зачисления.

**5.2 Сообщение оператору**

Тексты сообщений:

* "Успешный вход в систему."
* "Некорректные данные, пожалуйста, повторите ввод."
* "Заявка принята. Ожидайте результатов."
* "Заявка отклонена. Попробуйте подать новую заявку в следующем семестре."

Действия оператора:

* В случае успешного входа в систему: Продолжить выполнение необходимых операций.
* При вводе некорректных данных: Повторить ввод с корректными данными.
* После рассмотрения заявки: Принять необходимые меры согласно решению (принять, отклонить).

**6. Мероприятия по информационной безопасности**

**Шифрование данных:**

Применение алгоритмов шифрования для защиты конфиденциальных данных пользователей и результатов заявок.

**Контроль доступа:**

Реализация системы управления доступом, предоставляющей права только авторизованным пользователям в соответствии с их ролями (абитуриент, администратор).

**Регулярные аудиты безопасности:**

Проведение периодических аудитов безопасности для выявления и устранения потенциальных угроз и уязвимостей.

**Защита от SQL-инъекций:**

Применение параметризованных запросов для предотвращения SQL-инъекций при взаимодействии с базой данных.

**Мониторинг системных журналов:**

Организация системы мониторинга системных журналов для регистрации и анализа событий, связанных с безопасностью.

**Заключение**

"Система подачи заявок для абитуриентов" представляет собой комплексное решение, направленное на упрощение и улучшение процесса подачи заявок абитуриентами и эффективного управления ими со стороны администраторов.

В ходе разработки программы были решены различные задачи, начиная от проектирования базы данных, создания пользовательского интерфейса и заканчивая реализацией функционала обработки заявок. Внедрение системы позволит существенно оптимизировать взаимодействие абитуриентов и администраторов, сократив временные затраты и повысив общую эффективность.

Основные особенности разработанной системы включают в себя интуитивно понятный интерфейс, автоматизацию процессов обработки заявок, а также возможность оперативного взаимодействия между абитуриентами и администраторами.

В процессе работы были учтены и реализованы меры по обеспечению информационной безопасности, такие как шифрование данных, контроль доступа и регулярные аудиты. Это обеспечивает защиту конфиденциальности информации и предотвращение возможных угроз.

Однако, как и в любом IT-проекте, важно продолжать следить за тенденциями в сфере разработки программного обеспечения, а также учитывать обратную связь пользователей для последующего совершенствования системы.

В итоге проект успешно реализовал поставленные задачи, предоставляя эффективный инструмент для управления заявками абитуриентов и повышения качества взаимодействия в образовательном процессе.

**Приложения**

## Текст программы

*using* abiturient.ab;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace abiturient

{

public partial class mainab : Form

{

string name;

string login;

public mainab(string login)

{

InitializeComponent();

name = login;

label1.Text = login;

pbar.Value = 0;

textBox1.Visible = false;

button7.Visible = false;

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void mainab\_Load(object sender, EventArgs e)

{

}

public void ShowNotificationForm()

{

}

private void timer1\_Tick(object sender, EventArgs e)

{

pbar.Value += 1;

pbar.Text = pbar.Value.ToString()+"%";

if(pbar.Value == 89)

{

timer1.Enabled = false;

}

}

private void pbar\_Click(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button2\_Click(object sender, EventArgs e)

{

}

public void ShowNotificationForm(int userID)

{

// Показываем другую форму (замените FormToShow на ваш класс формы уведомлений)

settings sts = new settings(userID.ToString());

sts.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

prosmball bal = new prosmball(login);

this.Hide();

bal.Show();

}

private void button3\_Click(object sender, EventArgs e)

{

podachzayav pod = new podachzayav(login);

this.Hide();

pod.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

otslezayav ots = new otslezayav(login);

this.Hide();

ots.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

//settings ots = new settings(login);

//this.Hide();

//ots.Show();

}

private void button2\_Click\_1(object sender, EventArgs e)

{

otslezayav zv = new otslezayav(login);

this.Hide();

zv.Show();

}

private void button5\_Click\_1(object sender, EventArgs e)

{

textBox1.Visible = true;

button7.Visible = true;

}

private void textBox1\_Click(object sender, EventArgs e)

{

textBox1.Text = null;

}

private void button7\_Click(object sender, EventArgs e)

{

try

{

// Получаем введенное имя пользователя из TextBox

string usernameToCheck = textBox1.Text;

// Получаем UserID для введенного пользователя

int userId = GetUserIdByLogin(usernameToCheck);

// Проверяем зачисление пользователя на факультет

bool isAccepted = CheckAcceptStatus(userId);

// Выводим MessageBox с информацией

if (isAccepted)

{

string facultyName = GetFacultyNameByUserId(userId);

MessageBox.Show($"Абитуриент с именем {usernameToCheck} зачислен на факультет {facultyName}.");

}

else

{

MessageBox.Show($"Абитуриент с именем {usernameToCheck} не зачислен на факультет.");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при проверке зачисления: {ex.Message}");

}

}

private bool CheckAcceptStatus(int userId)

{

try

{

using (SqlConnection conn = new SqlConnection(connection.con))

{

conn.Open();

string query = "SELECT Accept FROM Applications WHERE UserID = @UserID";

using (SqlCommand command = new SqlCommand(query, conn))

{

command.Parameters.AddWithValue("@UserID", userId);

object result = command.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

return (bool)result;

}

// Если статус не найден, считаем, что абитуриент не зачислен

return false;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при проверке статуса зачисления: {ex.Message}");

return false;

}

}

private string GetFacultyNameByUserId(int userId)

{

try

{

using (SqlConnection conn = new SqlConnection(connection.con))

{

conn.Open();

string query = "SELECT F.FacultyName " +

"FROM Applications A " +

"INNER JOIN Faculties F ON A.FacultyID = F.FacultyID " +

"WHERE A.UserID = @UserID";

using (SqlCommand command = new SqlCommand(query, conn))

{

command.Parameters.AddWithValue("@UserID", userId);

object result = command.ExecuteScalar();

if (result != null && result != DBNull.Value)

{

return result.ToString();

}

// Если факультет не найден, возвращаем пустую строку

return string.Empty;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при получении имени факультета: {ex.Message}");

return string.Empty;

}

}

private int GetUserIdByLogin(string login)

{

try

{

using (SqlConnection conn = new SqlConnection(connection.con))

{

conn.Open();

string query = "SELECT UserID FROM Users WHERE Username = @Username";

using (SqlCommand command = new SqlCommand(query, conn))

{

command.Parameters.AddWithValue("@Username", login);

object result = command.ExecuteScalar();

if (result != null)

{

int userId = (int)result;

return userId;

}

// Возвращаем -1, если пользователь не найден

return -1;

}

}

}

catch (Exception ex)

{

// Обработка ошибки, например, вывод в консоль или логирование

Console.WriteLine($"Ошибка при получении UserID: {ex.Message}");

return -1;

}

}

}

}

**Источники, использованные при разработке**

1. ГОСТ Р ИСО\_МЭК 25051-2017 Требования к качеству готового к использованию программного продукта (RUSP) и инструкции по тестированию.
2. ЕСПД Единая система программной документации.
3. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов, Дэвид Скляр, 2017
4. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5, Никсон Робин, 2019
5. Изучаем JavaScript. Руководство по созданию современных веб-сайтов, Этан Браун, 2017

**Internet – ресурсы**

1. PHP: Hypertext preprocessor [Электронный ресурс] – [http://php.net](http://php.net/)
2. JavaScript Tutorial [Электронный ресурс] – <http://w3schools.com/js>
3. MySQL [Электронный ресурс] - <https://www.mysql.com/>
4. htmlbook.ru | Для тех кто делает сайты [Электронный ресурс] -http://htmlbook.ru/