Федеральное государственное образовательное бюджетное учреждение  
высшего образования

«Финансовый университет при Правительстве Российской Федерации»

КОЛЛЕДЖ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

**ОТЧЕТ**

**по результатам проектирования**

**и разработки программного приложения**

**по профессиональному модулю ПМ.03 Участие в интеграции программных модулей**

**специальности 09.02.03 Программирование в компьютерных системах**

Выполнил:

студент группы 4ПКС-220

Анисковцев Иван Анатольевич

Проверили:

преподаватели

\_\_\_\_\_\_\_\_ / Т.Г. Аксёнова /

\_\_\_\_\_\_\_\_/О.А. Тимакин/

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2023

Оглавление

[Проектирование программного приложения 3](#_Toc154589448)

[Руководство оператора 6](#_Toc154589449)

[Листинг программных модулей 10](#_Toc154589450)

# **Проектирование программного приложения**

На рисунке 1 представлена диаграмма «сущность-связь» предметной области «Калькулятор ипотечного кредитования»:

Изображение выглядит как текст, диаграмма, снимок экрана, Параллельный

Автоматически созданное описание

Рисунок 1. Диаграмма «сущность-связь»

На следующих рисунках находятся таблицы с заполненными данными. На рисунке 2 изображена таблица «Banks».

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 2. Таблица «Banks»

На рисунке 3 изображена таблица «Debtors».

Изображение выглядит как снимок экрана, текст, число, Шрифт

Автоматически созданное описание

Рисунок 3. Таблица «Debtors»

На рисунке 4 изображена таблица «InformationAboutLoans».

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

Рисунок 4. Таблица «InformationAboutLoans»

На рисунке 5 изображена таблица «Loans».

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 5. Таблица «Loans»

На рисунке 6 изображена таблица «RealEstateCategory».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 6. Таблица «RealEstateCategory»

На рисунке 7 изображена таблица «RealEstateObjects».

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 7. Таблица «RealEstateObjects»

На рисунке 8 изображена таблица «RealEstateType».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8. Таблица «RealEstateType»

На рисунке 8 изображена таблица «Users».

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 9. Таблица «Users»

На рисунке 10 находится диаграмма вариантов использования предметной области «Калькулятор ипотечного кредитования»:

Изображение выглядит как зарисовка, текст, рисунок, диаграмма

Автоматически созданное описание

Рисунок 10. Диаграмма вариантов использования

# **Руководство оператора**

Программа предназначена для помощи пользователям в оценке ипотечных кредитов, а также для планирования и расчёта процесса погашения задолженности по ипотеке. Основные цели и функции программы включают:

1. расчет ежемесячного платежа: пользователь может ввести основные параметры кредита, такие как сумма кредита, процентная ставка и срок кредита, чтобы определить размер ежемесячного платежа;
2. график погашения задолженности: программа может создавать график погашения задолженности, отображая, как меняются основной долг и проценты в каждом ежемесячном платеже. Это позволяет понять, как распределяется платеж между основным долгом и процентами;
3. рассмотрение различных сценариев: пользователь может провести анализ различных сценариев, варьируя параметры кредита, такие как сумма, срок, процентная ставка. Это помогает принять информированное решение о выборе наилучших условий кредита;
4. загашение задолженности досрочно: программа может также рассчитать выгоду от досрочного погашения задолженности ипотеки, позволяя пользователям оценить, как дополнительные платежи влияют на срок кредита и общие затраты;
5. отображение общей суммы к возврату: программа предоставляет пользователю общую сумму, которую необходимо вернуть банку по итогам кредита, включая основной долг и проценты.

Программа призвана обеспечить пользователей необходимой информацией для принятия обоснованных решений относительно ипотечного кредитования, а также для оптимизации процесса погашения задолженности, учитывая их финансовые возможности и цели.

Системные требования для работы программного продукта должны быть следующими:

- тактовая частота процессора -1 200 Гц;

- объем оперативной памяти 64 Мб; объем свободного дискового

- пространства 50 Мб; разрешение монитора 1 024 х 768;

- наличие устройства чтения компакт-дисков.

Программа должна работать в операционных системах:

- Windows 10, 11.

Для выполнения программы оператору требуется запустить программу и авторизоваться. Далее в зависимости от роли пользователь будет попадать на разные экраны. На рисунке 11 изображена страница авторизации.

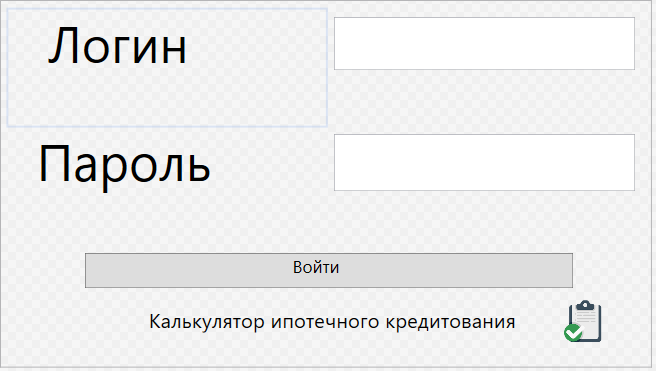


Рисунок 11. Страница авторизации

На рисунке 12 изображён экран для заёмщика на рисунке 13 изображён экран для банка.

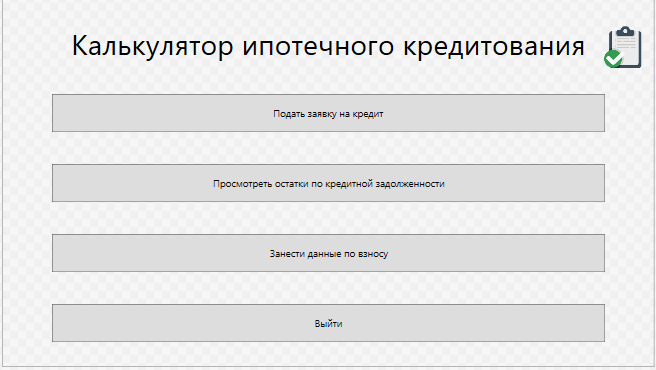


Рисунок 12. Экран заёмщика

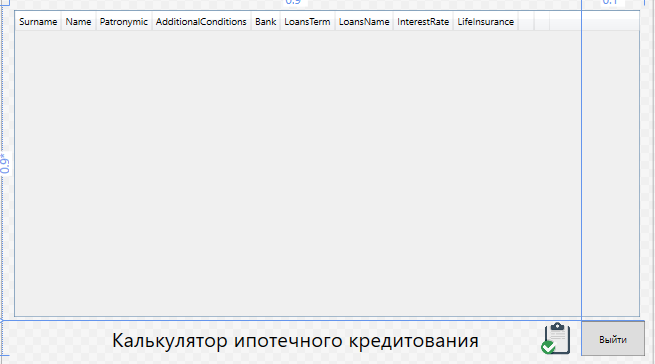


Рисунок 13. Экран сотрудника банка

Пользователь может подать заявку на кредит, посмотреть остатки по кредитной задолженности, занести данные по взносу.

На рисунке 14 изображён экран с взносом, на рисунке 15 изображён экран с остатками по кредиту.

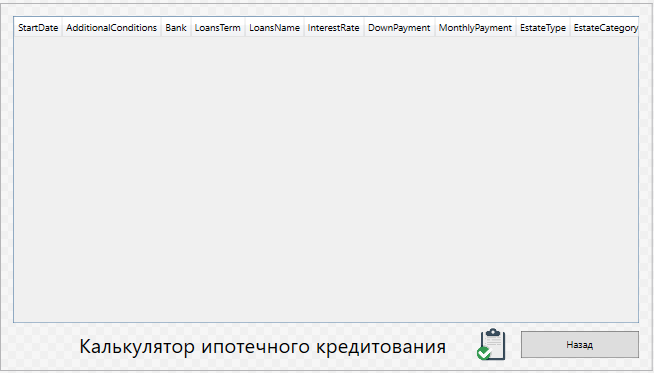


Рисунок 14. Экран с взносом

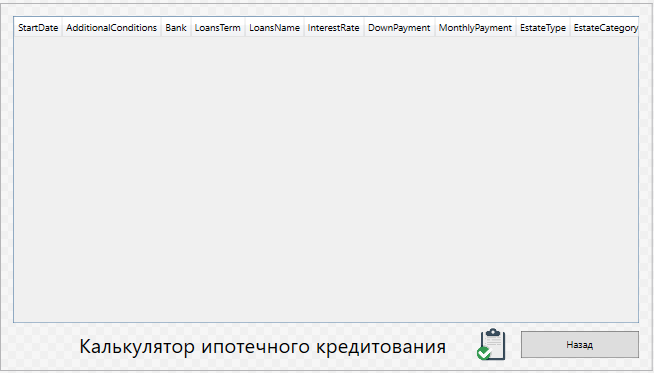


Рисунок 15. Экран с остатками по кредиту

# **Листинг программных модулей**

Листинг модуля подачи заявки:

namespace PR30.Pages

{

public partial class ApplyingForLoanPage : Page

{

private Dictionary<string, bool> \_lifeInsurance = new Dictionary<string, bool>()

{

["Да"] = true,

["Нет"] = false

};

private CreditsEntities \_db = new CreditsEntities();

private int \_debtorId;

public ApplyingForLoanPage(int debtorId)

{

InitializeComponent();

\_debtorId = debtorId;

Banks.ItemsSource = \_db.Banks.ToList();

LoansName.ItemsSource = new[] { "Кредит", "Ипотека" };

LifeInsurance.ItemsSource = \_lifeInsurance.Keys;

RealEstateType.ItemsSource = \_db.RealEstateTypes.ToList();

RealEstateCategory.ItemsSource = \_db.RealEstateCategories.ToList();

new List<ComboBox> { Banks, LoansName, LifeInsurance, RealEstateType, RealEstateCategory }.ForEach(cb => cb.SelectedIndex = 0);

}

~ApplyingForLoanPage()

{

\_db.Dispose();

}

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

{

try

{

if (!Int32.TryParse(LoansTerm.Text, out var loansTerm))

throw new Exception("Срок кредита должен быть числом");

if (loansTerm <= 0)

throw new Exception("Срок кредита должен быть больше или равен 1");

if (!Decimal.TryParse(InterestRate.Text, out var interestRate))

throw new Exception("Процентная ставка должна быть числом");

if (interestRate < 0)

throw new Exception("Процентная ставка не может быть отрицательна");

if (!Decimal.TryParse(DownPayment.Text, out var downPayment))

throw new Exception("Начальный платеж должен быть числом");

if (downPayment < 0)

throw new Exception("Начальный платеж не может быть отрицательным");

if (!Decimal.TryParse(Price.Text, out var price))

throw new Exception("Сумма кредита должна быть числом");

if (price <= 0)

throw new Exception("Сумма кредита не может быть меньше или равна 0");

if (downPayment >= price)

throw new Exception("Сумма начального платежа не может быть больше или равна сумме кредита");

decimal priceWithLoansTerm = price + price \* (100 + interestRate) / 100;

InformationAboutLoan informationAboutLoan = new InformationAboutLoan

{

LoansName = LoansName.Text,

LoansTerm = loansTerm,

InterestRate = interestRate,

DownPayment = downPayment,

MonthlyPayment = (priceWithLoansTerm - downPayment) / loansTerm,

LifeInsurance = \_lifeInsurance[LifeInsurance.Text]

};

\_db.InformationAboutLoans.Add(informationAboutLoan);

RealEstateObject realEstateObject = new RealEstateObject()

{

RealEstateTypeID = ((RealEstateType)RealEstateType.SelectedValue).ID,

RealEstateCategoryID = ((RealEstateCategory)RealEstateCategory.SelectedValue).ID,

Price = priceWithLoansTerm,

Address = String.IsNullOrWhiteSpace(Address.Text) ? null : Address.Text,

Paid = downPayment

};

\_db.RealEstateObjects.Add(realEstateObject);

Loan loan = new Loan

{

BankID = ((Bank)Banks.SelectedValue).ID,

StartDate = DateTime.Now,

DebtorID = \_debtorId,

InformationAboutLoansID = informationAboutLoan.ID,

RealEstateObjectsID = realEstateObject.ID,

AdditionalConditions = String.IsNullOrWhiteSpace(AdditionalConditions.Text) ? "Отсутствует" : AdditionalConditions.Text,

IsAccepted = false,

IsDecline = false,

IsPaid = false

};

\_db.Loans.Add(loan);

\_db.SaveChanges();

MessageBox.Show("Заявка успешно оформлена!");

NavigationService.GoBack();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

=> NavigationService.GoBack();

}

}

Листинг модуля авторизации:

public partial class AuthorizationPage : Page

{

private CreditsEntities \_db = new CreditsEntities();

public AuthorizationPage()

=> InitializeComponent();

~AuthorizationPage()

{

\_db.Dispose();

}

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

{

try

{

User user = \_db.Users.FirstOrDefault(u => u.Login == Login.Text && u.Password == Password.Password);

if (user is null) throw new System.Exception("Пользователь не найден");

MessageBox.Show($"Добро пожаловать, {user.Name} {user.Patronymic}");

Debtor debtor = \_db.Debtors.FirstOrDefault(d => d.UserID == user.ID);

if (debtor != null)

NavigationService.Navigate(new MainPageForDebtors(debtor.ID));

else

NavigationService.Navigate(new MainPageForBankEmployeer());

}

catch

{

MessageBox.Show("Некорректные данные");

}

}

}

Листинг модуля остатки по кредиту:

namespace PR30.Pages

{

public partial class ViewingLoanBalancesPage : Page

{

private CreditsEntities \_db = new CreditsEntities();

private int \_debtorId;

public ViewingLoanBalancesPage(int debtorId)

{

InitializeComponent();

\_debtorId = debtorId;

UpdateList();

}

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

=> NavigationService.GoBack();

private void UpdateList()

{

Loans.ItemsSource = \_db.Loans.Where(l => l.DebtorID == \_debtorId && l.IsAccepted == true && l.IsPaid == false).Select(l => new

{

l.StartDate,

l.InformationAboutLoan.DownPayment,

l.InformationAboutLoan.MonthlyPayment,

l.RealEstateObject.RealEstateType.EstateType,

l.RealEstateObject.RealEstateCategory.EstateCategory,

l.RealEstateObject.Price,

l.RealEstateObject.Address,

l.RealEstateObject.Paid,

l.AdditionalConditions,

Bank = l.Bank.BankName,

l.InformationAboutLoan.LoansTerm,

l.InformationAboutLoan.LoansName,

l.InformationAboutLoan.InterestRate,

l.InformationAboutLoan.LifeInsurance,

}).ToList();

}

}

}

Листинг модуля взноса:

namespace PR30.Pages

{

public partial class EnterDataOnContributionPage : Page

{

private CreditsEntities \_db = new CreditsEntities();

private int \_debtorId;

public EnterDataOnContributionPage(int debtorId)

{

InitializeComponent();

\_debtorId = debtorId;

UpdateList();

}

~EnterDataOnContributionPage()

{

\_db.Dispose();

}

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

=> NavigationService.GoBack();

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

int loanId = Int32.Parse(((Button)sender).Tag.ToString());

try

{

Loan loan = \_db.Loans.Find(loanId);

RealEstateObject reo = loan.RealEstateObject;

\_db.Entry(reo).State = EntityState.Modified;

reo.Paid += loan.InformationAboutLoan.MonthlyPayment;

if (reo.Paid >= reo.Price)

loan.IsPaid = true;

\_db.SaveChanges();

UpdateList();

MessageBox.Show("Месячный платеж внесен!");

if (loan.IsPaid.Value)

MessageBox.Show("Задолженность погашена!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void UpdateList()

{

Loans.ItemsSource = \_db.Loans.Where(l => l.DebtorID == \_debtorId && l.IsAccepted == true && l.IsPaid == false).Select(l => new

{

Id = l.ID,

l.StartDate,

l.InformationAboutLoan.DownPayment,

l.InformationAboutLoan.MonthlyPayment,

l.RealEstateObject.RealEstateType.EstateType,

l.RealEstateObject.RealEstateCategory.EstateCategory,

l.RealEstateObject.Price,

l.RealEstateObject.Address,

l.RealEstateObject.Paid,

l.AdditionalConditions,

Bank = l.Bank.BankName,

l.InformationAboutLoan.LoansTerm,

l.InformationAboutLoan.LoansName,

l.InformationAboutLoan.InterestRate,

l.InformationAboutLoan.LifeInsurance,

}).ToList();

}

}

}

Листинг модуля для заёмщиков:

namespace PR30.Pages

{

public partial class MainPageForDebtors : Page

{

private int \_debtorId;

public MainPageForDebtors(int debtorId)

{

InitializeComponent();

\_debtorId = debtorId;

}

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

=> NavigationService.Navigate(new ApplyingForLoanPage(\_debtorId));

private void Button\_Click\_1(object sender, RoutedEventArgs e)

=> NavigationService.Navigate(new ViewingLoanBalancesPage(\_debtorId));

private void Button\_Click\_2(object sender, RoutedEventArgs e)

=> NavigationService.Navigate(new EnterDataOnContributionPage(\_debtorId));

private void Button\_Click\_3(object sender, RoutedEventArgs e)

=> NavigationService.Navigate(new AuthorizationPage());

}

}

Листинг модуля для сотрудника банка:

namespace PR30.Pages

{

public partial class MainPageForBankEmployeer : Page

{

private CreditsEntities \_db = new CreditsEntities();

public MainPageForBankEmployeer()

{

InitializeComponent();

UpdateList();

}

~MainPageForBankEmployeer()

{

\_db.Dispose();

}

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

=> Click(Int32.Parse(((Button)sender).Tag.ToString()), "одобрен", true, false);

private void Button\_Click\_1(object sender, RoutedEventArgs e)

=> Click(Int32.Parse(((Button)sender).Tag.ToString()), "отклонен", false, true);

private void Click(int loanId, string message, bool isAccepted, bool isDecline)

{

var loan = \_db.Loans.Find(loanId);

\_db.Entry(loan).State = EntityState.Modified;

loan.IsAccepted = isAccepted;

loan.IsDecline = isDecline;

\_db.SaveChanges();

MessageBox.Show($"Кредит для {loan.Debtor.User.Surname} {loan.Debtor.User.Name} {loan.Debtor.User.Patronymic} {message}!");

UpdateList();

}

private void UpdateList()

{

Loans.ItemsSource = \_db.Loans.Where(l => new[] { l.IsDecline, l.IsAccepted, l.IsPaid }.All(p => p.Value == false)).Select(l => new

{

Id = l.ID,

DebtorSurname = l.Debtor.User.Surname,

DebtorName = l.Debtor.User.Name,

DebtorPatronymic = l.Debtor.User.Surname,

l.AdditionalConditions,

Bank = l.Bank.BankName,

l.InformationAboutLoan.LoansTerm,

l.InformationAboutLoan.LoansName,

l.InformationAboutLoan.InterestRate,

l.InformationAboutLoan.LifeInsurance

}).ToList();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

=> NavigationService.Navigate(new AuthorizationPage());

}

}