

Документация на проект Infinia

1. Въведение:

1. Кратко описание на проекта

Проектът "Infinia" представлява модерна уеб платформа, предназначена за оптимизация на банковите операции, разработена с помощта на ASP.NET Core MVC. Основната цел на приложението е да подобри потребителското изживяване и да увеличи ефективността на финансовите транзакции, предоставяйки интуитивен интерфейс и усъвършенствани функции за управление на клиентските сметки, обработка на заеми, както и аналитични инструменти за финансови отчети.

Включвайки иновации, като изкуствен интелект, системата предлага автоматизирана клиентска поддръжка, оценка на кредитния риск и прогнозиране на финансовите резултати, които са от помощ за банковите администратори и интегрирана чатбот система, която подпомага клиентското потребление, като предлага отговор при въпросите на клиентите. С акцент върху сигурността и надеждността, проектът осигурява на потребителите безопасна среда за извършване на банкови транзакции.

Платформата е проектирана с модулен подход, разделена на три основни компонента:

Управление на клиентски сметки - Позволява на потребителите да управляват своите банкови сметки, да извършват транзакции и да получават актуализации относно баланса на сметките си.

Обработка на заеми и ипотечни кредити - Осигурява на потребителите лесен достъп до информация и услуги, свързани с различни видове заеми и автоматичната им обработка.

Аналитика и отчитане на финансите - Предоставя подробни отчети и анализи, които подпомагат банковите администратори във вземането на информирани решения.

Проектът "Infinia" е разработен от екип от пет души в рамките на 25 дни и е насочен към осигуряване на иновативни решения за банковата индустрия,

насочени към подобряване на обслужването на клиентите и оптимизация на бизнес процесите.

2. Цели и предназначение на системата

Проектът "Infinia" има за цел да предостави иновационни решения за банковия сектор, които да отговорят на нуждите на потребителите и да подобрят общата ефективност на банковите операции. Основните цели и предназначения на системата включват:

Оптимизация на банковите услуги: Системата е проектирана да автоматизира и опрости процесите на управление на клиентски сметки, като предлага интуитивен интерфейс за извършване на транзакции, проверка на баланса и управление на персонализирани настройки.

Улеснена обработка на заеми: Чрез интегрирането на функции за кандидатстване и управление на заеми, "Infinia" осигурява на потребителите лесен достъп до информация и услуги, свързани с различни видове кредити, включително условия за одобрение и актуализации за статуса на кредита. Също така имаме автоматизирано месечно погасяване на заема както и месечна такса за потребление.

Анализ и отчетност: Платформата предоставя аналитични инструменти за изготвяне на финансови отчети и анализи, което позволява на банковите администратори да вземат информирани решения и да проследяват своите финансовите резултати на всеки един клон от банката. Тези инструменти се ползват с помощта на изкуствен интелект, за да изгадят точни финансови прогнози и анализи.

Подобрено клиентско обслужване: Интеграцията на изкуствен интелект и автоматизирани услуги за клиентска поддръжка цели да подобри взаимодействието между клиентите и банката, като осигурява бързи и ефективни отговори на запитвания и проблеми.

Сигурност и надеждност: Проектът акцентира на сигурността на потребителските данни и транзакции, като внедрява надеждни технологии за защита и шифроване, които гарантират безопасността на финансовата информация.

Системата "Infinia" е предназначена да бъде лесно адаптивна и разширяема, позволявайки на банките да внедряват нови функции и услуги, отговарящи на променящите се нужди на пазара и потребителите.

Системата е разработена с оглед скалируемост, което спомага разширяването и.

2. Архитектура на системата

1. Общ преглед на архитектурата

Архитектурата на системата "Infinia" е проектирана с оглед на модулност, мащабируемост и сигурност, което позволява ефективно управление на банковите операции и предоставяне на висококачествени услуги на клиентите. Общийят преглед на архитектурата включва следните ключови компоненти:

Модулна структура: Системата е разделена на три основни модула:

Управление на клиентски сметки: Този модул отговаря за обработката на клиентски данни, управление на сметки и извършване на транзакции.

Обработка на заеми и ипотечни кредити: Модулът предоставя функционалности за кандидатстване, одобрение и управление на различни видове кредити, осигурявайки лесен достъп до информация и статуса на заявлениета.

Финансови анализи и отчети: Този модул предлага инструменти за генериране на отчети и анализи на финансови данни.

Архитектура с клиент-сървър: Системата използва архитектура с клиент-сървър, при която клиентските приложения комуникират с централния сървър, обработващ бизнес логиката и данните. Това осигурява бърза и надеждна работа на системата, като позволява обработка на множество заявки в реално време.

База данни: Системата използва релационна база данни, която осигурява съхранение и управление на данни, свързани с клиентите, транзакциите и финансовите отчети. Базата данни е проектирана с акцент на сигурността и интегритета на данните.

Интеграция на изкуствен интелект: Интеграцията на AI технологии в системата позволява анализ на данни за кредитоспособност и предлагане на чатбот услуги. Използване на AI за предоставяне на финансови отчети и анализи.

Сигурност: Системата прилага най-добрите практики за сигурност, включително шифроване на данни, автентикация на потребители и защита от злонамерени атаки. Това осигурява безопасност и надеждност на всички операции и данни в системата.

С архитектура, проектирана за бъдещи разширения и иновации, "Infinia" цели да предостави на клиентите ефективни и надеждни банкови услуги, отговарящи на съвременните изисквания на финансния сектор.

2. Използвани технологии

Проектът използва множество съвременни технологии и инструменти, за да създаде ефективна и мащабируема система за управление на банкови операции. Основните технологии, използвани в проекта, включват:

ASP.NET Core MVC: Проектът е изграден на базата на ASP.NET Core MVC, модерен и мощен уеб фреймворк за разработка на уеб приложения с моделно-изгледно-контролерна (MVC) архитектура. Това осигурява ясно разделение на логиката, лесна мащабируемост и добра поддръжка на тестването.

AI интеграции: Включването на изкуствен интелект (AI) в проекта се използва за подобряване на клиентските услуги, анализ на кредитоспособността, управление на риска и прогнозиране на финансови тенденции. Тези AI функции позволяват по-добро обслужване на клиентите и повишаване на ефективността на процесите. Интеграцията на AI моделите с нашето MVC приложение става чрез Flask API-та, към които ние пращаме заявки и получаваме отговор от тренираните модели, които обработваме и визуализираме на потребителя.

MailKit и MimeKit: За изпращане на имейли в системата се използват MailKit и MimeKit – библиотеки, които осигуряват мощни инструменти за изпращане на електронни съобщения чрез SMTP (например Gmail). Те се използват за комуникация с клиентите, включително за потвърждение на регистрация и при смяна на парола.

HTML, CSS и JavaScript: Фронтенд частта на системата е изградена с помощта на HTML, CSS и JavaScript. Тези технологии се използват за изграждане на потребителския интерфейс, като се грижат за структуриране на съдържанието, стиловете и интерактивността. Те са интегрирани

директно в Razor синтаксиса на ASP.NET Core, което позволява динамично и сигурно създаване на изгледи и ефективно взаимодействие с бекенда.

Entity Framework Core: За работа с базата данни се използва Entity Framework Core – ORM (Object-Relational Mapping) инструмент, който улеснява работата с релационни бази данни като SQL Server. Това позволява на разработчиците да използват обектно-ориентиран подход за работа с данни, без директно да пишат SQL заявки.

NUnit: За тестване на функционалностите на системата се използва NUnit – библиотека за писане на модулни тестове, която осигурява стабилност и увереност в качеството на приложението. Модулните тестове покриват около 80% от бизнес логиката.

Тези технологии, комбинирани в рамките на един проект, осигуряват висока производителност, сигурност и възможност за мащабиране на системата, като същевременно предоставят интуитивен интерфейс за крайните потребители.

3.Структура на проектите

Описание на методите в **AccountController**

CreateAccount

Този метод обработва POST заявка за създаване на нов акаунт. Проверява валидността на входните данни и извършва създаването на акаунта.

```
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> CreateAccount(CreateAccountViewModel model)
{
    if (ModelState.IsValid == false)
    {
        return View(model);
    }
    if (model.Balance < 0)
    {
        ModelState.AddModelError("Balance", NegativeBalanceErrorMessage);
        return View(model);
    }
    var userId = User.GetId();
    if (await profileService.CustomerWithIdExistsAsync(userId) == false)
    {
        return BadRequest();
    }
    if (await profileService.CustomerWithIdentityCardNumberExists(model.IdentityCardNumber, userId) == false)
    {
        ModelState.AddModelError("IdentityCardNumber", InvalidIdentityCardNumberErrorMessage);
        return View(model);
    }
    await accountService.CreateAccountAsync(model, userId);
    return RedirectToAction(nameof(Index));
}
```

DeleteAccount

Методът обработва заявката за изтриване на акаунт по предоставеното id. Уведомява потребителя за успеха или неуспеха на операцията.

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> DeleteConfirmed(int id, DeleteAccountViewModel model)
{
    if (await accountService.AccountWithIdExistsAsync(id) == false)
    {
        return BadRequest();
    }
    await accountService.DeleteAccountAsync(model.Id);
    return RedirectToAction(nameof(Index));
}
```

AccountDetails

Методът връща информация за конкретен акаунт, идентифициран по id. Той предоставя детайли, свързани с акаунта, като баланси и транзакции.

```
[HttpGet]
0 references
public async Task<IActionResult> Details(int id)
{
    if (await accountService.AccountWithIdExistsAsync(id) == false)
    {
        return BadRequest();
    }
    var model = await accountService.GetAccountDetailsAsync(id);
    return View(model);
}
```

Описания на методите в **LoanController**:

ApplyForLoan

Този метод обработва POST заявката за кандидатстване за заем. Проверява валидността на входните данни и, ако всичко е наред, обработва заявката за заем.

```

public async Task<IActionResult> ApplyForLoan(LoanApplicationViewModel model)
{
    if (ModelState.IsValid == false)
    {
        return View(model);
    }

    var userId = User.GetId();
    if (await profileService.CustomerWithIdExistsAsync(userId) == false)
    {
        return BadRequest();
    }
    if (await profileService.CustomerWithIdentityCardExists(model, userId) == false)
    {
        ModelState.AddModelError("SSN", InavlidCustomerIdentityCardErrorMessage);
        return View(model);
    }
    if (await profileService.CustomerWithAddressExists(model, userId) == false)
    {
        ModelState.AddModelError("Country", InavlidCustomerAddressErrorMessage);
        return View(model);
    }
    if (await profileService.CustomerWithAccountIBANExists(model.AccountIBAN, userId) == false)
    {
        ModelState.AddModelError("AccountIBAN", InavlidCustomerAccountIBANErrorMessage);
        return View(model);
    }
    await loanService.ApplyForLoanAsync(model, userId);

    return RedirectToAction(nameof(LoanApplicationsHistory));
}

```

CurrentLoans

Методът връща информация за текущите заеми на потребителя. Той показва списък с активните заеми и техните детайли.

```

[HttpGet]
0 references
public async Task<IActionResult> CurrentLoans()
{
    var userId = User.GetId();
    if (await profileService.CustomerWithIdExistsAsync(userId) == false)
    {
        return BadRequest();
    }
    var model = await loanService.GetCurrentLoansForCustomerAsync(userId);
    return View(model);
}

```

LoanApplicationsHistory

Методът предоставя история на подадените заявления за заем. Потребителят може да види статуса и детайлите на предишните си заявки.

```
[HttpGet]
1 reference
public async Task<IActionResult> LoanApplicationsHistory()
{
    var userId = User.GetId();
    if (await profileService.CustomerWithIdExistsAsync(userId) == false)
    {
        return BadRequest();
    }
    var model = await loanService.GetLoanApplicationHistoryForCustomerAsync(userId);
    return View(model);
}
```

Описания на методите в **ProfileController**:

Login

Този метод обработва POST заявката за вход на потребителя. Проверява валидността на данните и извършва вход, ако всичко е наред, или показва съобщения за грешки.

```
var passwordCheck = await userManager.CheckPasswordAsync(user, model.Password);
if (passwordCheck == false)
{
    ModelState.AddModelError("Password", InvalidPasswordErrorMessage);
    return View(model);
}
if (await userManager.IsEmailConfirmedAsync(user) == false)
{
    ModelState.AddModelError("", EmailNotConfirmedErrorMessage);
    return View(model);
}
var result = await signInManager.PasswordSignInAsync(user, model.Password, false, false);
if (result.Succeeded)
{
    return RedirectToAction("Index", "Home");
}
```

ProfileDetails

Методът показва детайлите на профила на текущия потребител. Извлича информация от услугата за профил и я предава на изгледа.

```
[HttpGet]
2 references
public async Task<IActionResult> ProfileDetails()
{
    var userId = User.GetId();
    if(await profileService.CustomerWithIdExistsAsync(userId)) == false)
    {
        return BadRequest();
    }
    var model = await profileService.GetProfileDetailsAsync(userId);
    return View(model);
}

[HttpPost]
```

ConfirmEmail

Този метод потвърдява имейл адреса на потребителя с предоставен идентификатор и токен. Ако потвърждението е успешно, потребителят е пренасочен към страницата за вход.

```
1 reference
public async Task<IActionResult> ConfirmEmail(string userId, string token)
{
    if (userId == null || token == null)
    {
        return BadRequest();
    }
    var user = await userManager.FindByIdAsync(userId);
    if (user == null)
    {
        return BadRequest();
    }
    var result = await userManager.ConfirmEmailAsync(user, token);
    if (result.Succeeded)
    {
        return RedirectToAction(nameof(Login));
    }
    return View();
}
```

ForgotPassword

Този метод обработва POST заявката за забравена парола. Проверява валидността на данните и, ако всичко е наред, генерира нова парола и я изпраща на имейл адреса на потребителя.

```
var newPassword = GenerateRandomPassword();

var token = await userManager.GeneratePasswordResetTokenAsync(user);
var result = await userManager.ResetPasswordAsync(user, token, newPassword);

if (result.Succeeded)
{
    var emailBody = $"<h1>Password Reset</h1><p>Your new password is: <strong>{newPassword}</strong></p>";
    await emailSenderService.SendEmailAsync(model.Email, "Password Reset", emailBody);

    TempData["ConfirmationMessage"] = "A new password has been sent to your email.";
    return RedirectToAction(nameof(Login));
}
```

Register

Този метод обработва POST заявката за регистрация на нов потребител. Проверява валидността на данните и, ако всичко е наред, създава нов потребител и изпраща имейл за потвърждение.

```
var result = await userManager.CreateAsync(user, model.Password);
if (result.Succeeded)
{
    var token = await userManager.GenerateEmailConfirmationTokenAsync(user);
    var confirmationLink = Url.Action(nameof(ConfirmEmail), "Profile", new { userId = user.Id, token = token }, Request.Scheme);

    var emailBody = $"<h1>Confirm Your Email</h1><p>Please confirm your email by clicking the link: <a href='{confirmationLink}'>Confirm Email</a></p>";
    await emailSenderService.SendEmailAsync(model.Email, "Email Confirmation", emailBody);
    TempData["ConfirmationMessage"] = "A confirmation email has been sent to your registered email address. Please check your inbox.";
    return RedirectToAction("RegistrationConfirmed");
}
```

Описания на методите в **TransactionController**:

MakeTransactionBetweenMyAccounts

Този метод обработва POST заявката за извършване на транзакция между акаунти. Проверява валидността на данните, проверява наличността на средства и извършва транзакцията.

MakeTransactionWithinTheBank

Този метод обработва POST заявката за транзакция в рамките на банката. Проверява валидността на данните, проверява наличността на средства и извършва транзакцията.

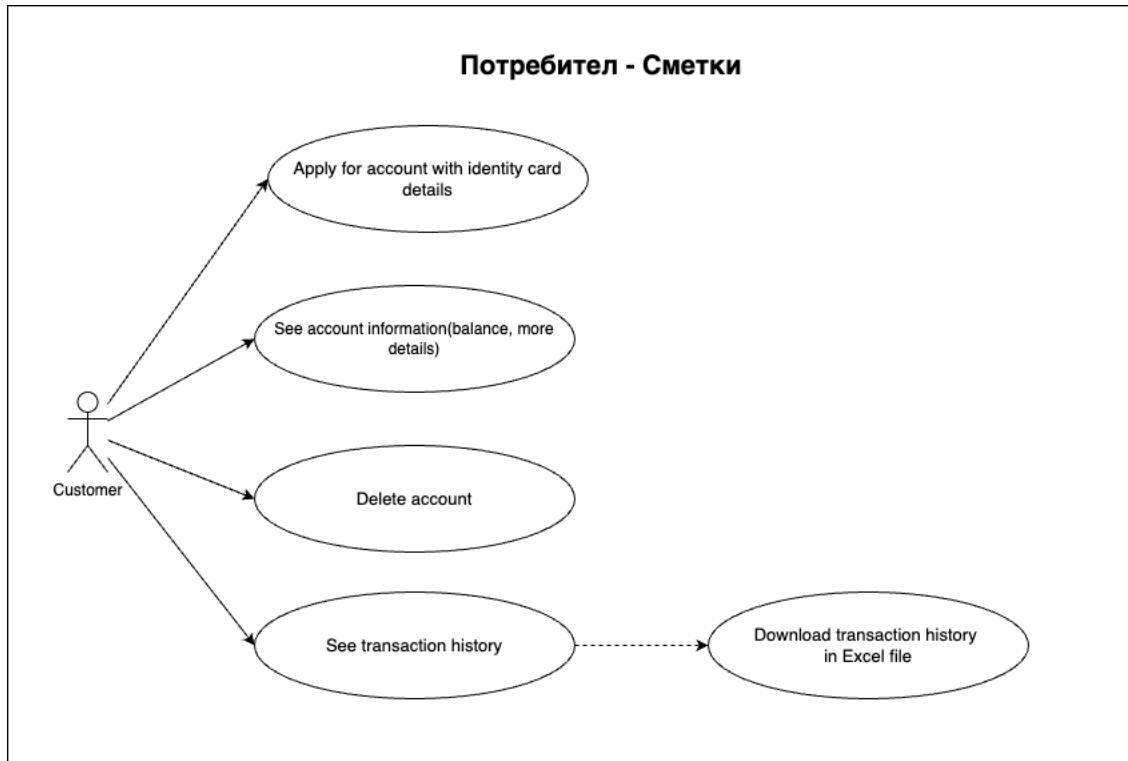
MakeTransactionToAnotherBank

Този метод обработва POST заявката за извършване на транзакция към друга банка. Проверява валидността на данните, проверява наличността на средства и извършва транзакцията.

DownloadTransactionHistoryForAccount

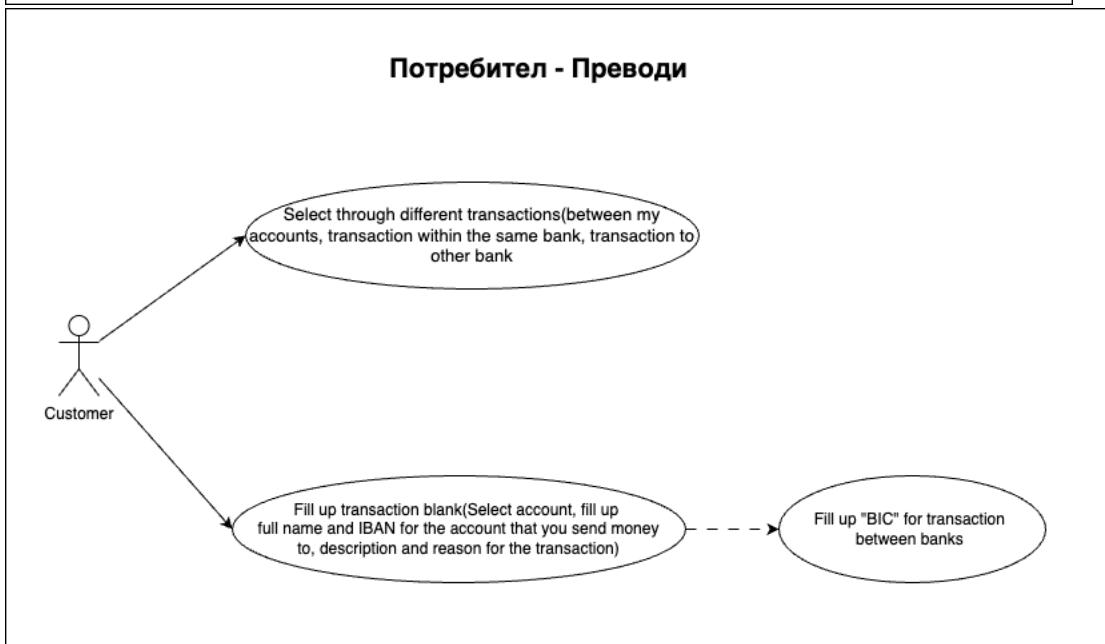
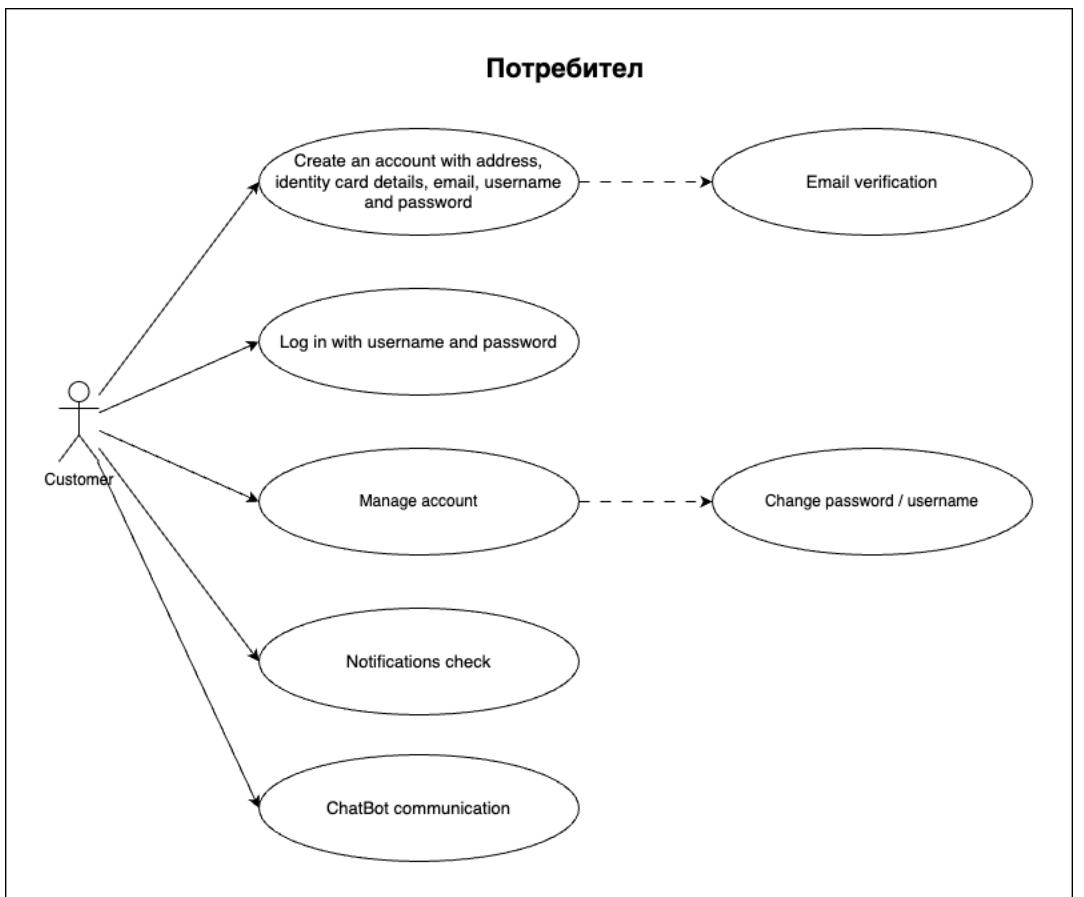
Методът генерира и предоставя CSV файл с история на транзакциите за конкретен акаунт. Проверява дали акаунтът съществува и след това извършва необходимите операции за генериране на файла.

4. Class диаграми



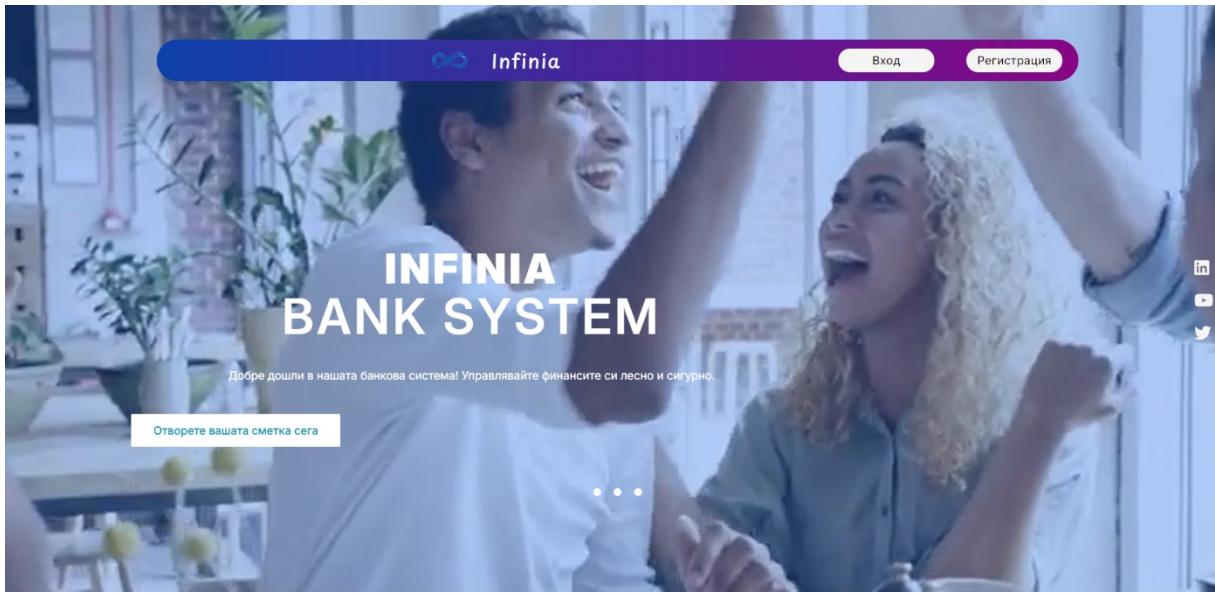
Потребител - Кредити





3. Модули на приложението

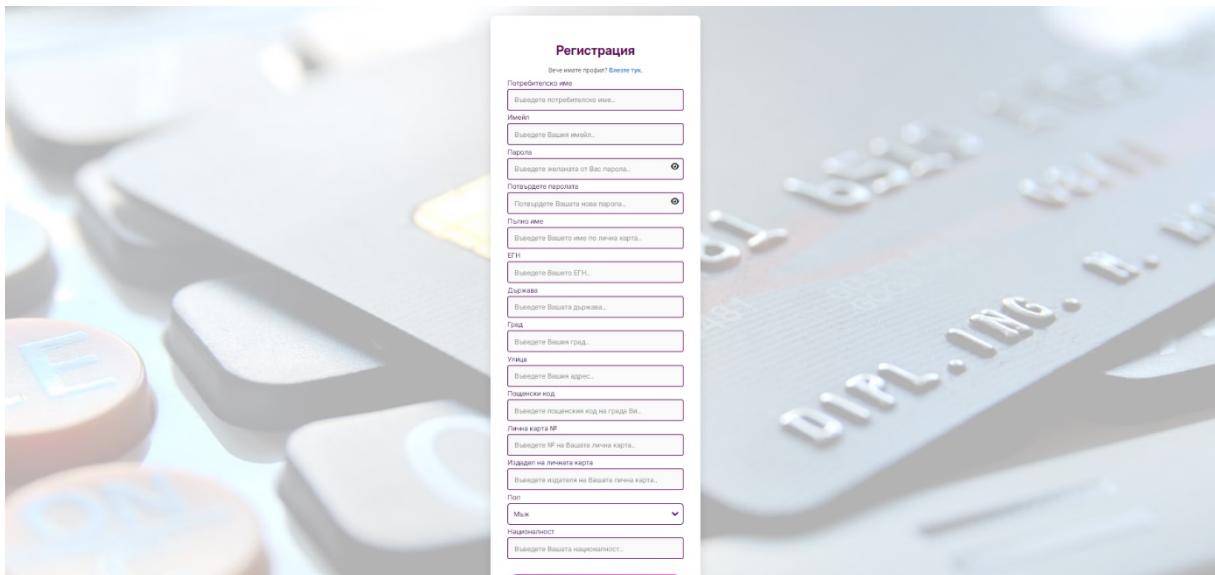
1. Начална страница за нерегистрирани потребители



На началната страница имаме бутони за регистрация и вход.

След като натиснем бутона за регистрация се зарежда форма за попълване на информация, нужна за регистрирането на потребителя.

2. Управление на клиентските акаунти



След като потребителят е събмитнал формата ще му излезе съобщение за потвърждение на имейл.

Успешна регистрация

Изпратен е имейл за потвърждение на въведения от Вас имейл адрес!

Моля, проверете своята входяща кутия и кликнете на линка за потвърждение, за да активирате профила си.

Ако не получите имейл от нас до няколко минути, проверете папката "Спам".

© 2024 Infinia created by Albiceleste

След потвърждение на имейла ни препраща към страницата за вход, където да въведем нашите username и парола.

Вход

Потребителско име

Въведете Вашето потребителско име..

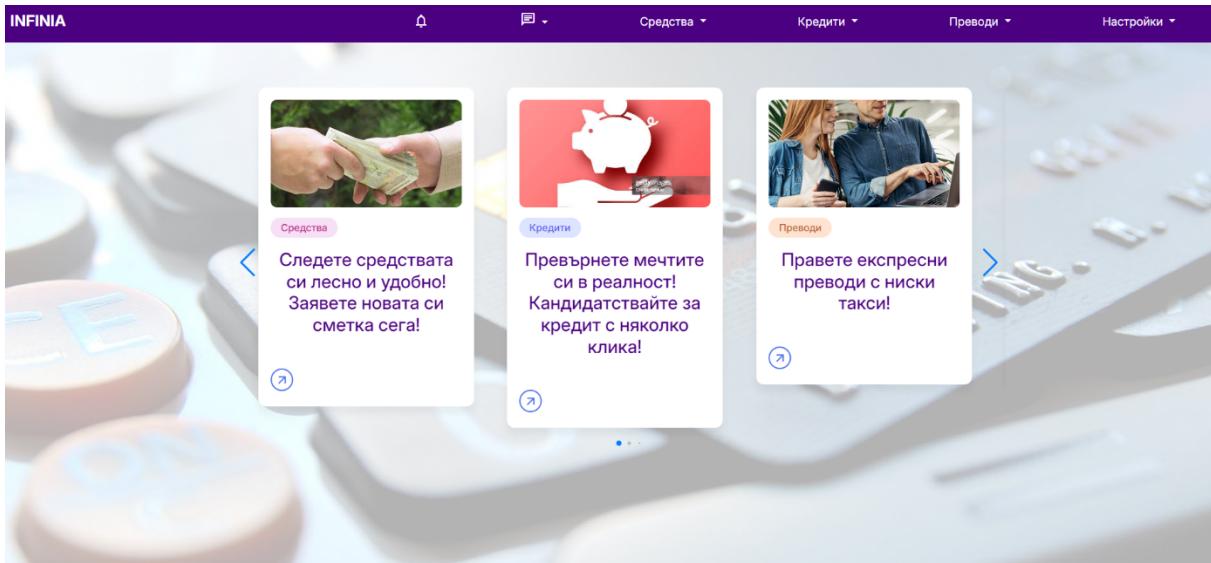
Парола

Въведете Вашата парола..

[Forgot Password?](#)

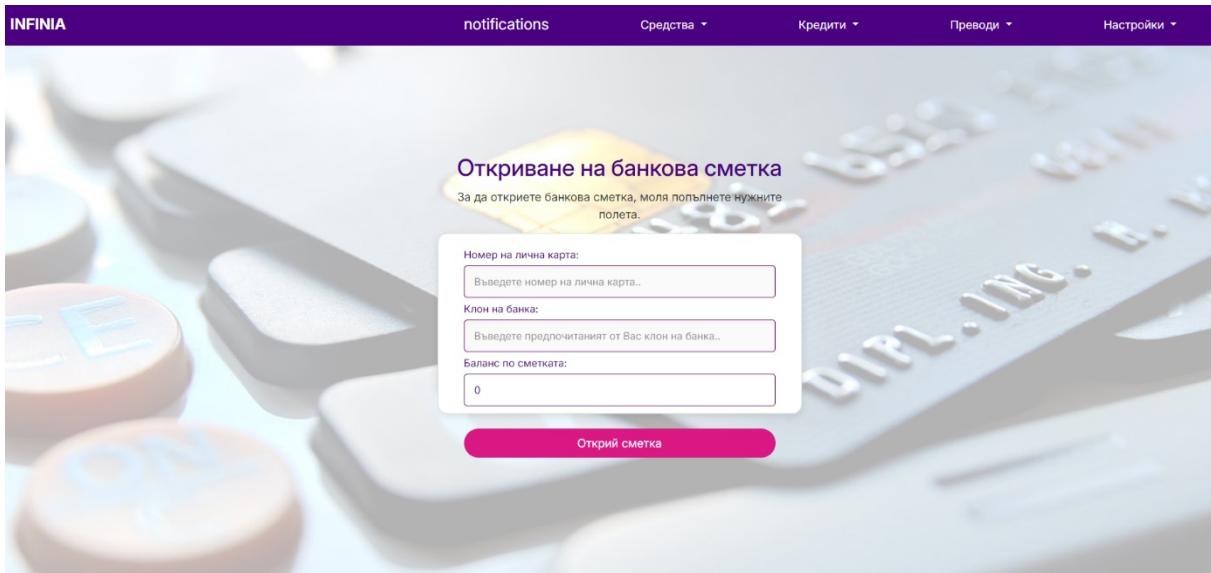
Login

След успешно влизане в акаунта ни препраща на началната страница, където можем да видим всички различни функционалности на нашата апликация като те са: Бутон към нашите сметки и средства, Бутон към кредити(Заявка на кредит, Активни кредити и История на заявка за кредити),Бутон за преводи, който ни предоставя възможността за различни видове преводи,Бутон за настройки на профила, Бутон за нотификации и Бутон за чатбот.

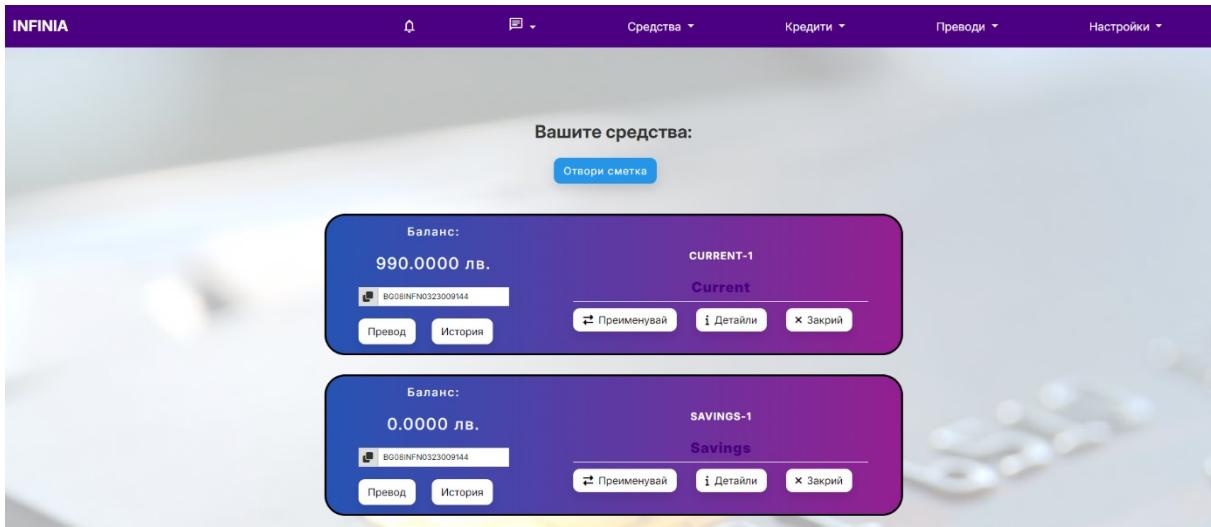


3. Сметки на потребителя

Когато натиснем бутона за откриване на сметка ни препраща на тази страница:



След като открием сметка ни препраща на страницата със сметки:



Там имаме възможност да преименуваме сметката, да видим детайли за акаунта и да я закрием. Също така можем да достъпим функционалността за превод.

История на транзакции:

The screenshot shows the transaction history section of the INFINIA mobile banking app. At the top, there is a purple header bar with the INFINIA logo, a menu icon, and several dropdown menus for 'Средства' (Means), 'Кредити' (Credits), 'Преводи' (Transfers), and 'Настройки' (Settings). Below the header, the main content area has a blurred background image of a keyboard and coins. In the center, there is a red button labeled 'Изтегли' (Download). A section titled 'История на транзакциите:' (Transaction history) is shown with a table:

Дата	Основание и пояснение	Наредител/Получител	Сума
25.09.2024	Зареждане И Зареждане	BG03INFN0480614677	10.00 лв.

At the bottom of the table, there is a blue button labeled 'Към сметки' (To account).

Детайли по сметка:

INFINIA

Средства

Кредити

Преводи

Настройки

Детайли за банкова сметка:

Име: CURRENT-1
IBAN: ---BG08INFN0323009144---
Баланс по сметката: 990.00 лв.
Открыта в клон: Sliven
Открыта на: 24.09.2024
Статус: Открыта

Към сметки

Закриване на сметка:

INFINIA

Средства

Кредити

Преводи

Настройки

Сигурни ли сте, че искате да закриете сметката си?

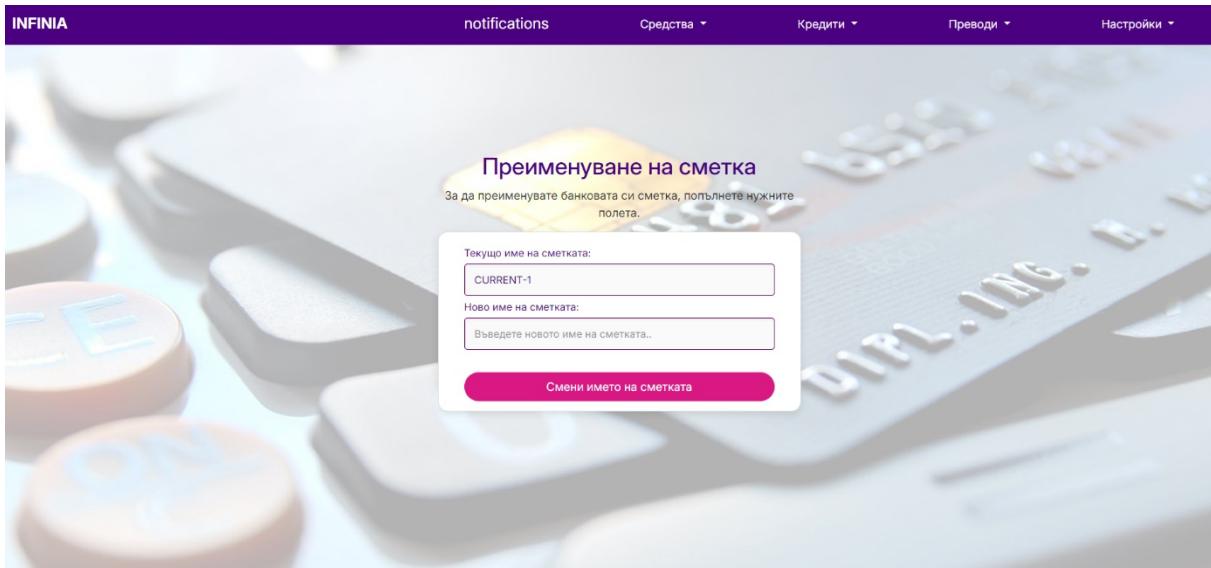
Не можете да изтриете тази сметка.

Данни за сметка:

Тип на сметка: CURRENT-1
IBAN на сметка: ---BG08INFN0323009144---
Баланс по сметка: 990.0000 лв.
Спестовна сметка: SAVINGS-1
IBAN на сметка: ---BG08INFN0323009144---
Баланс по сметка: 0.0000 лв.

Към сметки

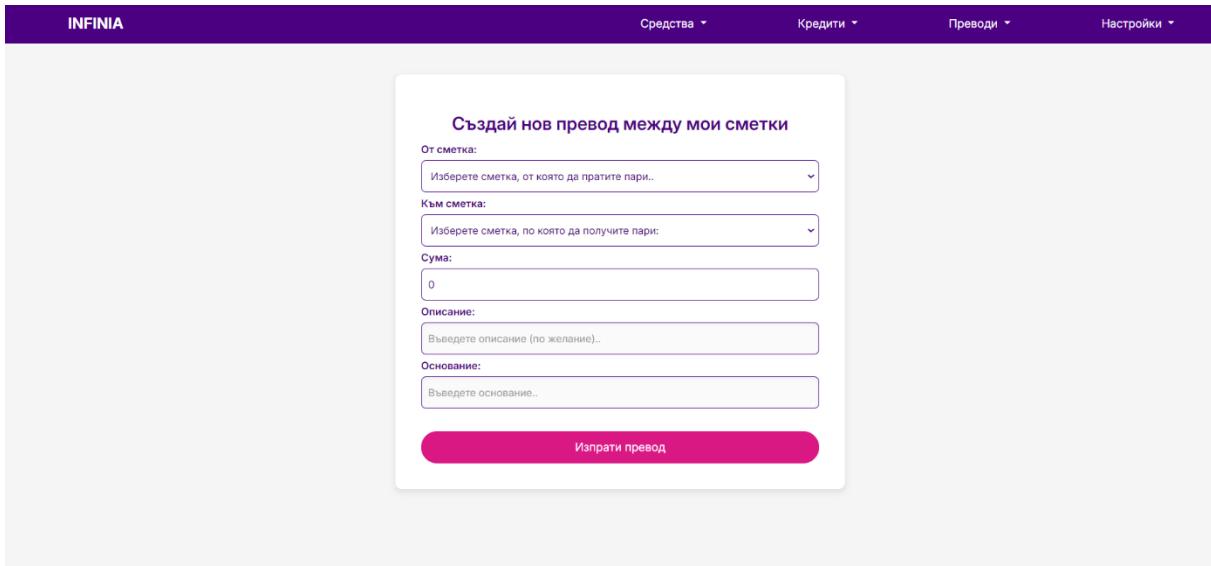
Преименуване на сметка:



4. Транзакции на потребителя(преводи)

Когато натиснем бутона преводи се зарежда страница с различните видове преводи, които са възможни.

Ако изберем „Между мои сметки“ се зарежда форма за съответния превод.



Ако изберем „По сметка“ се зарежда форма за съответния превод.

INFINIA

Средства ▾ Кредити ▾ Преводи ▾ Настройки ▾

Нареждане за превод по сметка в Infinia

От сметка:

Изберете сметка:

Име на получател:

Въведете пълното име на получателя..

IBAN на получател:

Въведете IBAN на получателя..

Сума:

0.00

Описание:

Въведете описание (по желание)..

Основание:

Въведете основание..

Изпрати превод

Ако изберем „Междубанков“ се зарежда форма за съответния превод.

INFINIA

Средства ▾ Кредити ▾ Преводи ▾ Настройки ▾

Нареждане за междубанков превод

От сметка:

Изберете сметка, от която да пратите пари..

Име на получател:

Въведете пълното име на получателя..

IBAN на получателя:

Въведете IBAN на получателя..

Сума:

0.00

BIC:

Въведете BIC на банката на получателя..

Описание:

Въведете описание (по желание)..

Основание:

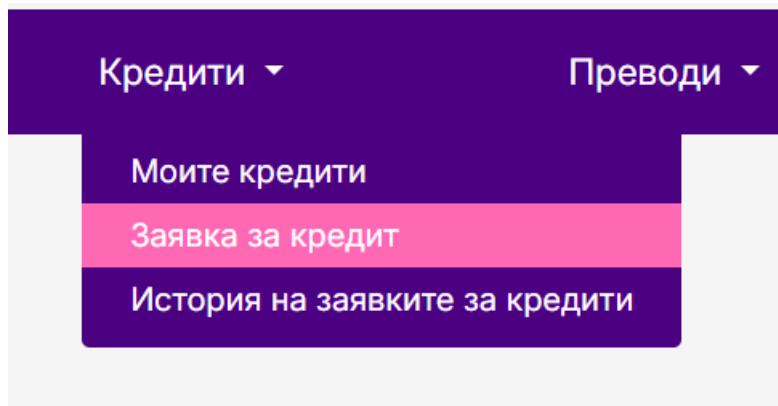
Въведете основание..

Изпрати превод

Моля, попълнете това поле.

5. Заеми

От падащото меню избираме бутон Заявка за кредит при което се зареждат различните видове кредити, които можем да изтеглим.



INFINIA

Средства ▾ Кредити ▾ Преводи ▾ Настройки ▾

Видове кредити и лихви

- Потребителски кредит Лихва: 8% [Заяви кредит](#)
- Авто кредит Лихва: 4% [Заяви кредит](#)
- Ипотечен кредит Лихва: 7% [Заяви кредит](#)
- Бизнес кредит Лихва: 5% [Заяви кредит](#)
- Кредит за образование Лихва: 3% [Заяви кредит](#)

След като изберем даден кредит се зарежда форма за попълване на лични данни и информация на потребителя.

INFINIA

Средства ▾ Кредити ▾ Преводи ▾ Настройки ▾

Заявка за кредит

1. Лични данни

ЕГН:
Въведете Вашето ЕГН..

Лична карта №:
Въведете № на Вашата лична карта..

Лична карта издадена на (**дата **месец ****год)
Въведете датата, на която е издадена личната Ви карта..

Издадел на личната карта
Въведете издателя на Вашата лична карта..

Пол
Въведете Вашия пол

Националност
Въведете Вашата националност

Държава

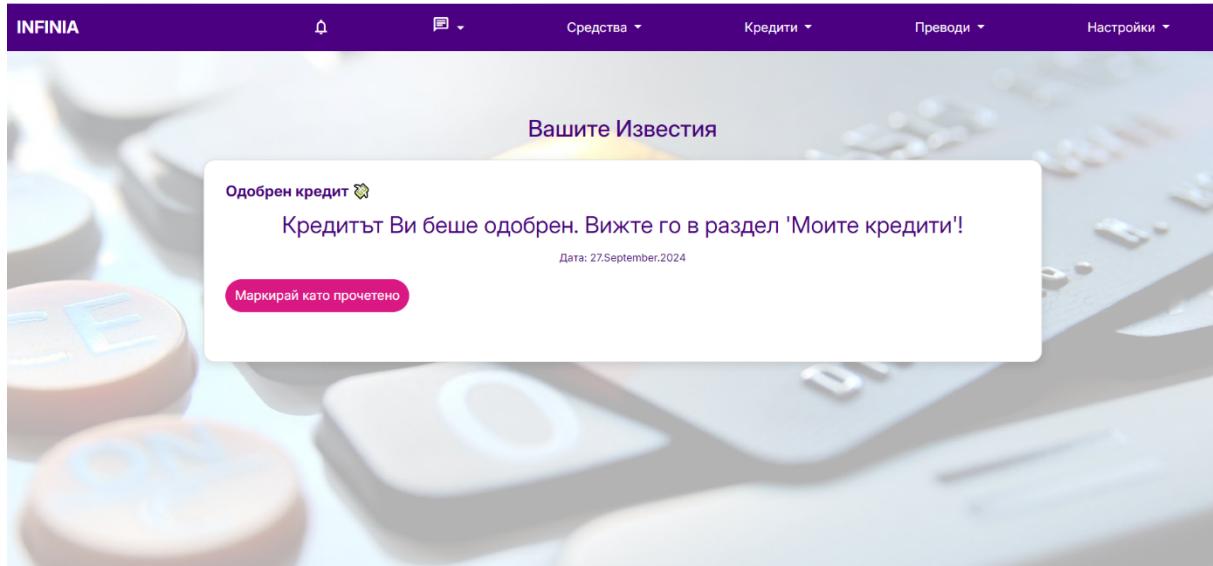
След вече изпращане на формата клиентът чака одобрение от банката.

Имаме бутона „Моите кредити“, който показва настоящите кредити към дадения потребител.

Също така имаме бутона за история на заявките за кредити.

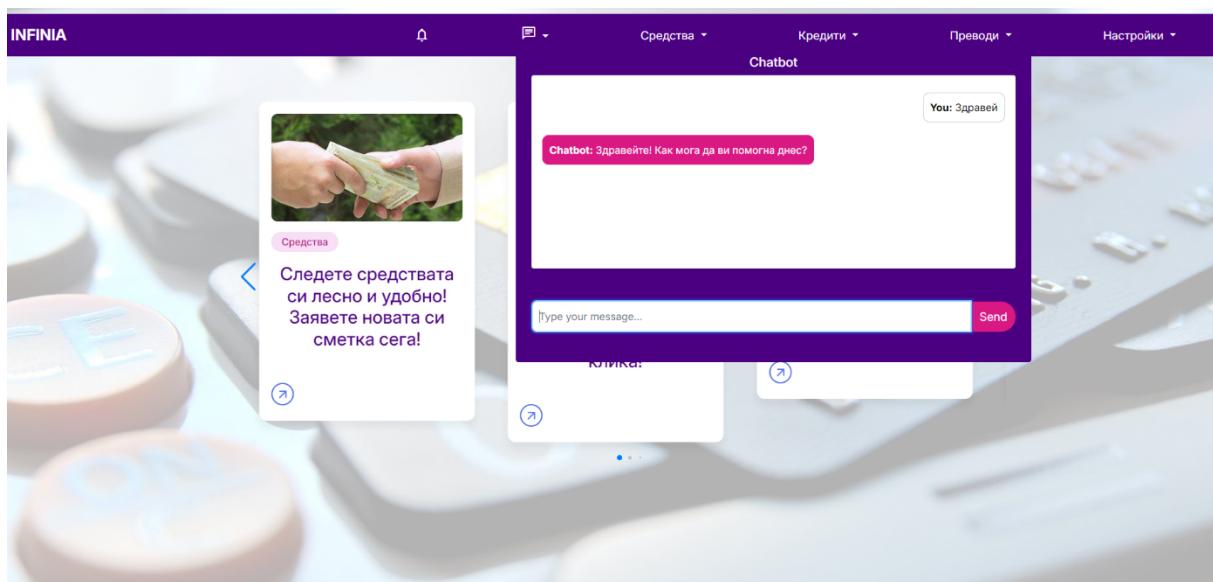
6. Нотификации

Когато натиснем бутона за нотификации ни изпраща на страница където виждаме всички съобщения.

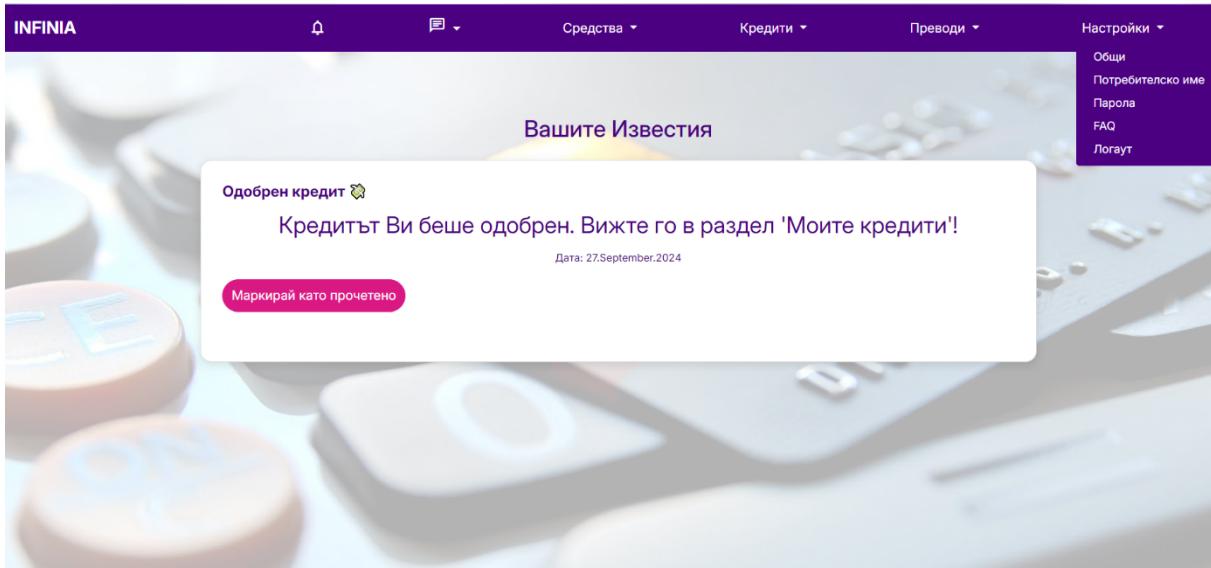


7. Чатбот

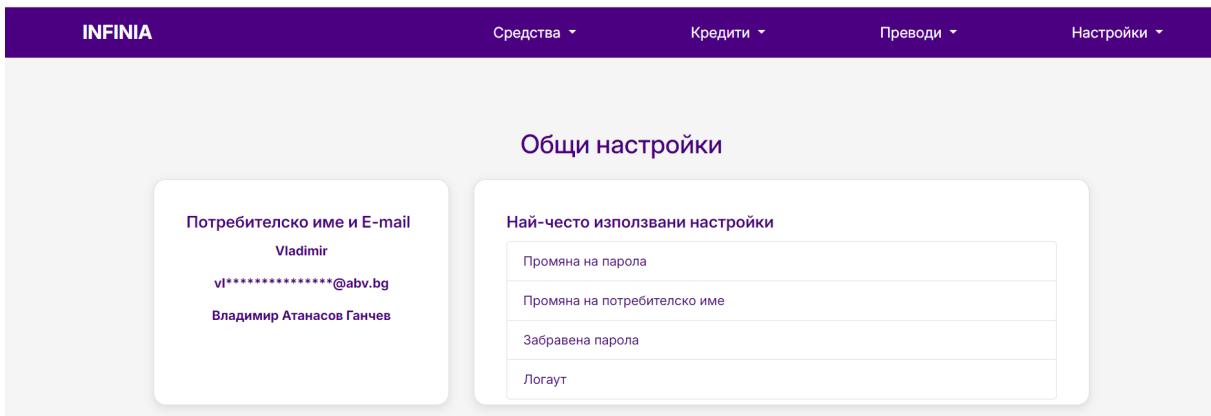
Когато натиснем бутона за чатбот се зарежда чатбота:



8. Настройки



Когато натиснем бутона Настройки се появява падащо меню с различни опции, а именно те са: Общи настройки, които ни показват информация за профила и FAQ.



Смяна на Потребителско име, което ни зарежда форма за въвеждане на старото потребителско име и желаното ново.

Смяна на потребителско име

За да смените потребителското име, попълнете нужните полета.

Текущо потребителско име:

Въведете текущото потребителско име

Ново потребителско име:

Въведете новото потребителско име

Потвърдете новото потребителско име:

Потвърдете новото потребителско име

Смени потребителското име

Смяна на Парола, което ни зарежда форма за въвеждане на старата парола и желаната нова.

Смяна на потребителско име

За да смените потребителското име, попълнете нужните полета.

Текущо потребителско име:

Въведете текущото потребителско име

Ново потребителско име:

Въведете новото потребителско име

Потвърдете новото потребителско име:

Потвърдете новото потребителско име

Смени потребителското име