НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ

ФАКУЛТЕТ ЗА БАЗОВО ОБРАЗОВАНИЕ

Програма „ИНФОРМАЦИОННИ ТЕХНОЛОГИИ”

ПРОЕКТ "ПОДГОТОВКА НА СПЕЦИАЛИСТИ ПО ИНФОРМАЦИОННИ ТЕХНОЛОГИИ В ИКОНОМИКАТА НА ЗНАНИЕТО‘‘

КУРС CITB613 ПРАКТИКА ПО БИЗНЕС ИНФОРМАЦИОННИ СИСТЕМИ

***ТЕМА*:** СИСТЕМА ЗА ПРЕДОСТАВЯНЕ НА БАНКОВИ ПРОДУКТИ И СРАВНЕНИЕ НА ДЕПОЗИТИ

Разработили:

Емилия Стоянова F108556

Христо Георгиев F108832

Димо Димчев F108582

Ръководител:

гл. ас. д-р Л. Томов

София, 2025 г.

СЪДЪРЖАНИЕ

1. Увод

Настоящият проект, изготвен във връзка с курс ‘’CITB613 Практика по бизнес информационни системи‘‘, има за цел да представи реализация на логика за изчисление на депозитен разплащателен план. За тази цел по темата на проекта са изготвени документ с пояснявane на изискванията и тяхното разпределение към съответните компоненти на системата, дизайн на потребителския интерфейс за уеб сайта на системата, който представя както десктоп резолюция, така и мобилна, имплементация на сървърна, клиентска логика, база данни, тестови сюити с ръчни и автоматични тестове. Крайният продукт е система, която представя иновативен и модерен интерфейс с опциите за разглеждане на различни банкови продукти, намиране на депозит, чрез набор от филтри – с опциите за разширено и за съкратено търсене – и изчисление на разплащателен план за избран депозит, според неговите условия и зададена сума, валута и период.

Ролите в екипа бяха разпределени по следния начин:

- Димо и Христо – архитекти и софтуерни инженери
- Емилия – дизайнер и тестер.

Ролята на експерт по документацията беше разделена между членовете на екипа, тъй като подлежаха на общо тълкование и събиране в един документ.

2. Презентация на системата

Системата за банкови продукти се състои от два основни компонента – десктоп приложение и уеб сайт. След стартиране на съответните сървъри се установява достъп до десктоп приложението. Неговата задача е да предостави интерфейс за създаване на депозити, както и на други банкови продукти. Тъй като целта на заданието е да се имплементира логика само за създаване и изчисление на депозитни планове, създаването на банкови продукти, различни от депозити, не е обвързано с логика и изчисления – то е само компонент на потребителския интерфейс. Създаването на депозит се осъществява чрез попълване на съответните полета в диалога. След избиране на ‘’Product Type: Deposit’’ се предоставят необходимите полета за създаването на депозита. Тези полета са:

- Име на депозита
- Минимална сума
- Период на депозита
- Валута
- Вида лице, за което е депозита
- Вид на депозита
- Вид на лихвата
- Периодичност на изплащане на лихвата
- Разрешаване на оувърдрафт

След натискане на бутона „Add” новият депозит е създаден и се визуализира в дясната част на диалога, която представлява списък на наличните банкови продукти.

Уеб сайтът се състои от три основни страници – уводна страница, страница за намиране на депозит и страница за пресмятане на разплащателен план. Уводната страница има за цел да представи различните банкови продукти, включително и депозитите. Всеки банков продукт е представен в компонент на страницата, който има дроп-даун опция, чрез която да се предостави допълнителна информация за съответния продукт, като такава информация е налична само за депозитните продукти. Навигационният бар в горната част на началната страница е наличен във всяка страница на сайта – в него има линкове към началната страница и към тази за търсене на депозити. Страницата за търсене на депозити предоставя два вида филтрация за намиране на конкретна депозитна оферта – разширена и съкратена, по подразбиране е заложена съкратената. В съкратената филтрация са включени следните опции:

- Избор за вид на депозита – срочен/безсрочен/спестовен
- Избор на валута – BGN, EUR, USD
- Минимална сума
- Максимална сума

- Срок – месеци на разплащателния план

Разширената филтрация добавя опциите:

- Тип лихва – фиксирана/променлива
- Позволяване на оувърдрафт – да/не

След натискане на бутона за търсене ще се покаже съответният резултат, ако съществува такъв, който да отговаря на критериите за търсене. Ако не бъдат въведени никакви филтри, то системата ще покаже всички създадени депозити.

Всяка от депозитните оферти, резултат от направеното търсене, предоставя детайлната информация за депозита , както и бутон за изчисление. След натискане на бутона за изчисление, потребителят е навигиран към страница за пресмятане на разплащателния план спрямо избрана оферта.

3. Начин на реализация

Както вече беше споменато, системата се състои от два основни компонента – десктоп приложение и уеб сайт. Десктоп приложението е реализирано с сървърна част BankAPI и DepositApp, където се стартира самата апликация.

BankAPI директорията се състои от основните компоненти за един .NET Web API проект, които са:

- • BankingAPI.csproj - Основният проектен файл на C# (.NET)
- • Program.cs - Главният входен файл за конфигурация и стартиране на приложението
- • DTOs, Models, Services, Validators - Поддиректории с логика и структури:
- DTOs - Data Transfer Objects (за обмен на данни между клиент и сървър)
- Models - Основните модели (например клиент, депозит и т.н.)
- Services - Бизнес логиката
- Validators - Валидация на входни данни

- • Data - Съдържа конфигурация за базата данни
- • appsettings.json - Конфигурационен файл
- • banking.db-shm, banking.db-wal - Части от SQLite база

  BankingAPI също може да се представи и като RESTful API, реализирано с помощта на ASP.NET Core, предназначено за управление на банкови депозити и банки. То служи като бекенд за комуникация с уеб и десктоп клиентите. Използваните технологии в него за C# като език за програмиране, ASP.NET Core Web API за фреймуърк, ORM – Entity Framework Core, SQLite за база данни, валидации с FLuentValidation и документация със Swagger. Тук е важно да разгледаме Program.cs файлът. Той конфигурира следните основни компоненти:

- Регистриране на DbContext със SQLite връзка
- Регистриране на услуги за валидация
- Добавяне на CORS политика (разрешение на заявки от всякакъв тип, което е необходимо за фронтенда)
- Регистрация на Swagger UI – достъпно в development среда
- Регистрация на бизнес логика чрез DepositCalculatorService

Основните REST API ендпойнти са:

- GET (/deposits) – връща списък с депозити
- GET (/deposits/{id} – връща конкретен депозит
- POST (/deposits) – създава нов депозит
- GET (/deposits/{id}/calculate?principal=X&durationMonths=Y) – изчислява очакваната печалба за даден депозит

Бизнес логиката в този компонент се реализира чрез услугата DepositCalculatorService, която служи за пресмятане на доходност от депозити. Това се случва на базата на въведени начална сума, продължителност в месеци и лихвен процент от избрания депозит.

DepositApp е десктоп клиент, разработен с помощта на Windows Forms (WinForms) — част от .NET платформата. Приложението служи за взаимодействие с потребителя във връзка с логиката за депозити (най-вероятно чрез UI, визуализиращ списъци, калкулатори и/или формуляри). Тук отново използваният език за програмиране е C#, фреймуъркът е .NET (WinForms), UI библиотека – Windows Forms. Входната точка на този компонент е методът Main() в Program.cs файлът. В него се използва ApplicationConfiguration.Initialize() метод, който настройва DPI скалирането, визуалния стил и други WinForms UI настройки. Стартирането на основната форма Form1 се осъществява чрез Application.Run(new Form1()). Тя предоставя на потребителя възможност за визуализация и създаване на банкови депозити чрез графичен интерфейс, който включва текстови полета, падащи менюта и списъци. Чрез вграден HttpClient, приложението осъществява HTTP заявки към локално стартиран API (на адрес http://localhost:5122/), като извлича наличните депозити и ги показва в списък. Освен това, потребителят може да създаде нов депозит чрез попълване на формата, след което данните се изпращат като POST заявка до API-то. Форма Form1 динамично адаптира изгледа си спрямо избора на тип продукт ("Deposit" или "Other") и съдържа вградено управление на валидации, enum стойности и визуални настройки. По този начин десктоп клиентът служи като пълноценен интерфейс за крайния потребител, чрез който се осъществява CRUD логиката за работа с депозити, дефинирана в сървърната част.

Фронтенд частта на приложението е изградена с помощта на React (версия 19) и JavaScript **(**ES6+**)**, с модулна структура и ясен компонентен подход. За стилове и UI дизайн се използва **TailwindCSS**, съчетан с компоненти от библиотеката **shadcn/ui**, което осигурява модерен, адаптивен и добре структуриран потребителски интерфейс. Визуалните компоненти са допълнително обогатени чрез CSS променливи и custom Tailwind слоеве, дефинирани в index.css, което позволява фина настройка на светла и тъмна тема, както и на типографията и spacing логиката.

Навигацията между отделните изгледи е реализирана чрез **React Router**, като главният компонент App.jsx дефинира маршрути към трите основни страници на приложението: MainPage, DepositSearch и DepositCalculator. За всяка от страниците се използват отделни компоненти, което гарантира висока степен на капсулация и повторна употреба на код. Общите

елементи като навигационната лента (Navbar) са изведени в самостоятелни компоненти, зареждани глобално чрез <Router> контейнера.

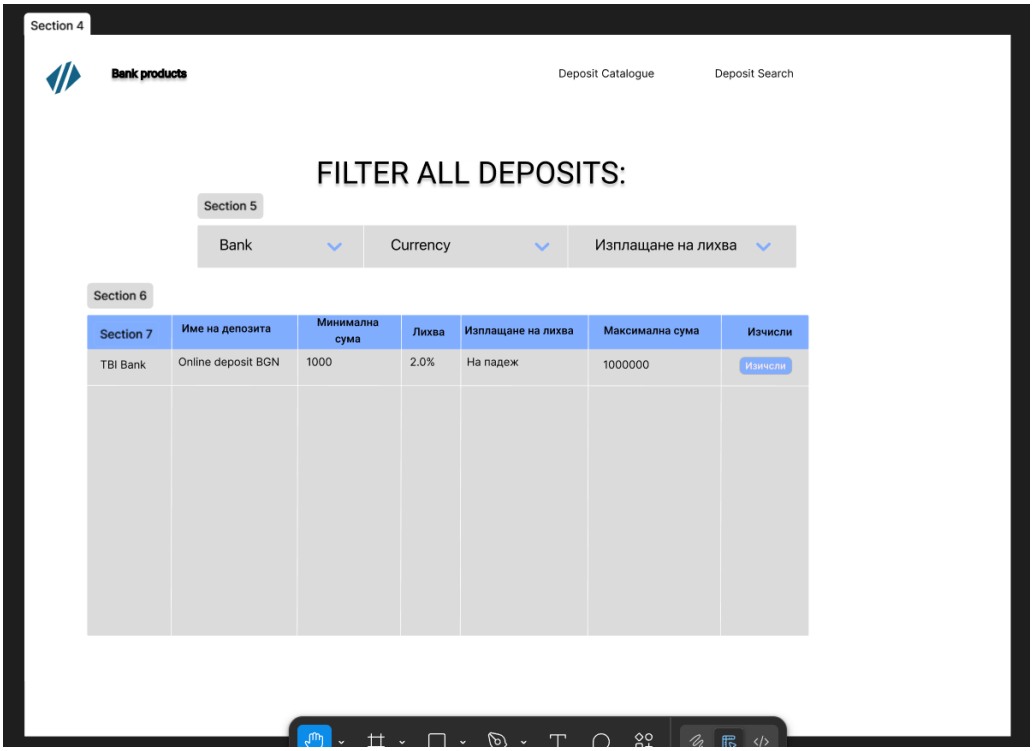
Функционалната логика във всеки компонент е реализирана с помощта на **React Hooks** – основно useState за управление на локално състояние (например стойности на формуляри, резултати от заявки) и useEffect за странични ефекти като извличане на данни от бекенда. За комуникация с REST API-то на бекенда се използва **Axios**, като заявките са асинхронни и интегрирани директно в логиката на компонентите – най-вече в Deposit-related функционалности.

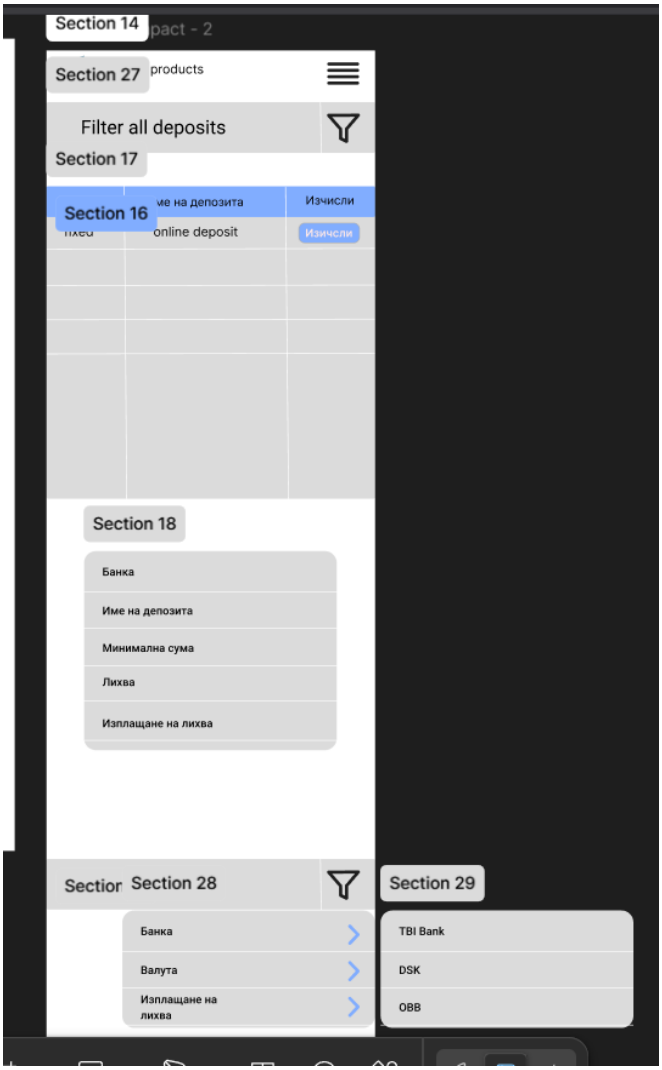
Формите за въвеждане и търсене на депозити включват валидиране на входните данни, базирано на JavaScript логика вътре в съответните компоненти, осигурявайки правилно структуриран JSON за изпращане към сървърната страна. Въпреки че валидирането е базово, структурата му позволява лесно разширение с по-строга логика или външни библиотеки.

Проектът използва **Vite** за бърза разработка и билдване, а tailwind.config и postcss.config (макар и не показани тук) вероятно управляват персонализацията на стиловата система. Присъствието на eslint и TypeScript типове подсказва намерение за поддържане на високо качество на кода.

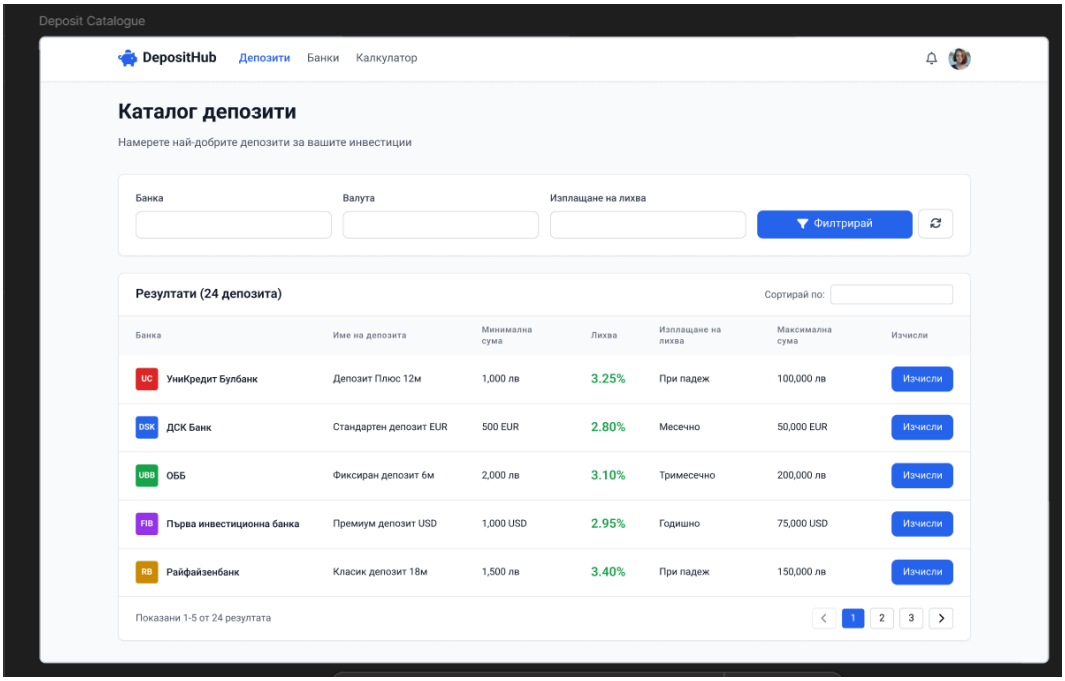
4. Дизайн

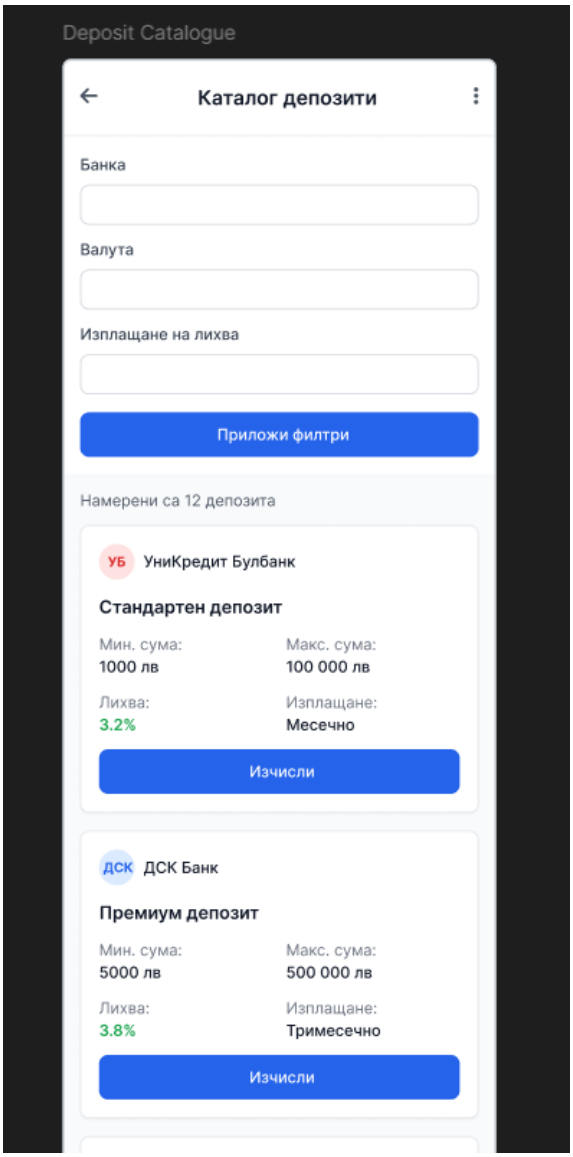
С цел по-лесна и бърза реализация на системата, всички се съгласихме, че е необходим дизайн само на уеб частта на системата. Дизайнът е изготвен в платформата Figma. Неготово създаване се базира на функционалните изисквания към проекта, описани в курса в Moodle, функционалностите и UI-а на уеб сайтът https://www.moitepari.bg/ , който ни беше предоставен за референция в документът с изисквания. Първо беше направен прототип на дизайна, чрез ръчно създаване на екраните и елементите в него. За всеки десктоп екран е създадена и мобилна версия, както и страница за съобщение за грешка тип 404. За пример, нека разгледаме страницата каталог на депозитите:


Section 4
Bank products
Deposit Catalogue
Deposit Search
FILTER ALL DEPOSITS:
Section 5
Bank
Currency
Изплащане на лихва
Section 6
Section 7
Име на депозита
Минимална сума
Лихва
Изплащане на лихва
Максимална сума
Изчисли
TBI Bank
Online deposit BGN
1000
2.0%
На падеж
1000000
Изчисли


Section 14
Section 27
products
Filter all deposits
Section 17
Section 16
Изчисли
online deposit
Изчисли
Section 18
Банка
Име на депозита
Минимална сума
Лихва
Изплащане на лихва
Section 28
Банка
Валута
Изплащане на лихва
Section 29
TBI Bank
DSK
OBB

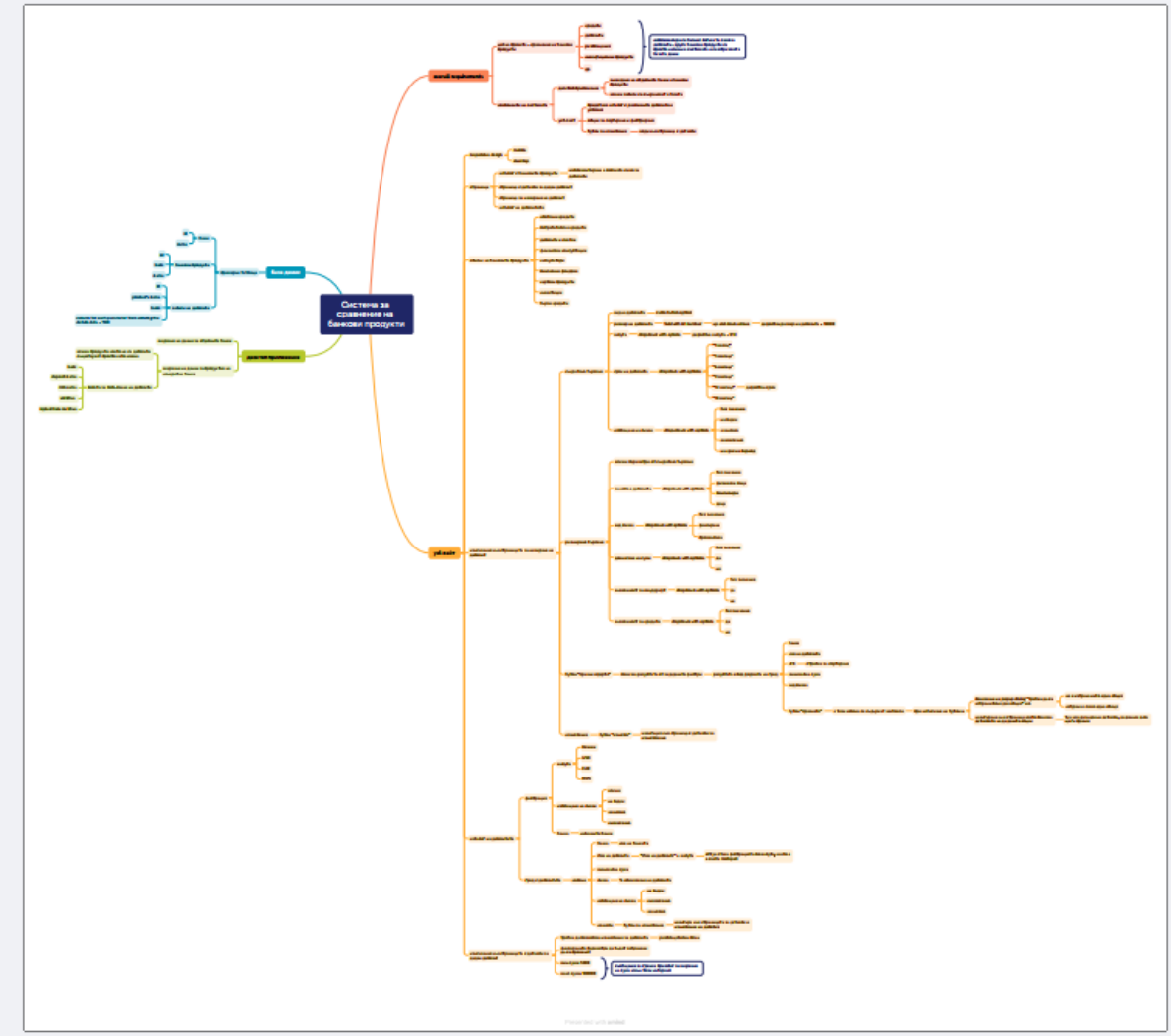
След като беше изготвен дизайнът на всички необходими страници се състоя среща, в която всички да го разгледаме и да преценим дали отразява необходимите функционалности и покрива критериите на проекта. За да се направи по иновативен и модерен дизайн, беше използван UX Pilot плъгина във Figma – представлява AI инструмент, с който да генерираме бързо и лесно дизайн на потребителския интерфейс от по-високо ниво, след като вече знаехме какви точно компоненти трябва да съдържа. След обработка с този плъгин, горната страница вече има този вид:


Deposit Catalogue
DepositHub
Депозити
Банки
Калкулатор
Каталог депозити
Намерете най-добрите депозити за вашите инвестиции
Банка
Валута
Изплащане на лихва
Филтрирай
Резултати (24 депозита)
Сортирай по:
Банка
Име на депозита
Минимална сума
Лихва
Изплащане на лихва
Максимална сума
Изчисли
УниКредит Булбанк
Депозит Плюс 12м
1,000 лв
3.25%
При падеж
100,000 лв
Изчисли
ДСК Банк
Стандартен депозит EUR
500 EUR
2.80%
Месечно
50,000 EUR
Изчисли
ОББ
Фиксиран депозит 6м
2,000 лв
3.10%
Тримесечно
200,000 лв
Изчисли
Първа инвестиционна банка
Премиум депозит USD
1,000 USD
2.95%
Годишно
75,000 USD
Изчисли
Райфайзенбанк
Класик депозит 18м
1,500 лв
3.40%
При падеж
150,000 лв
Изчисли
Показани 1-5 от 24 резултата
1
2
3


Deposit Catalogue
Каталог депозити
Банка
Валута
Изплащане на лихва
Приложи филтри
Намерени са 12 депозита
УБ
УниКредит Булбанк
Стандартен депозит
Мин. сума:
1000 лв
Макс. сума:
100 000 лв
Лихва:
3.2%
Изплащане:
Месечно
Изчисли
ДСК
ДСК Банк
Премиум депозит
Мин. сума:
5000 лв
Макс. сума:
500 000 лв
Лихва:
3.8%
Изплащане:
Тримесечно
Изчисли

5. Проследяване на изисквания

С цел по-добро разбиране на изискванията и разпределението им по различните компоненти беше направен майнд мап, въз основа за документа с изисквания в Moodle, както и логиката в сайта https://www.moitepari.bg/ . Файлът има следния вид:



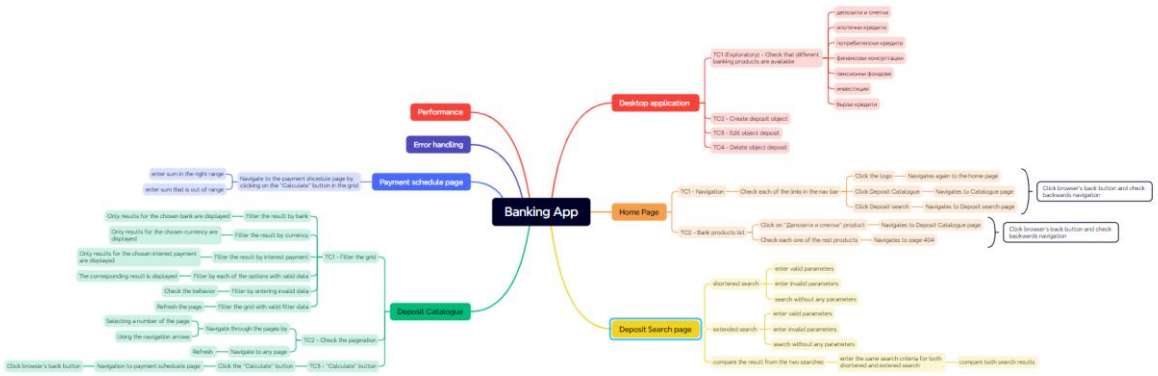
Всеки компонент е изнесен като тема на проекта, откъдето произлизат различните функционалности и изисквания към него.

6. Тестване

Тестването на системата се осъществява чрез комбинация от ръчно и автоматично тестване. Общата тестова документация включва – тестов план, тест дизайн (състоящ се от майнд мап и тестови сюити) и тестови скриптове.

Тестовият план въвежда начина на провеждане на тестването. Съдържа обща информация за проекта, обхват на тестовете, видове тестване, което ще бъде изпълнено, а именно – функционално, тестване на сигурността, тестване на производителността, натовареността и тестване на базата данни. Описани са критериите, които трябва да бъдат достигнати за да се счита качеството на проекта като приемливо, както и различните роли по изпълнение на тестовете и периодите за тях.

Тест дизайнът представлява документацията по ръчните тестове. Започнат е с условно разпределяне на тестовите компоненти по частите на системата. Майн мапът има следния вид:



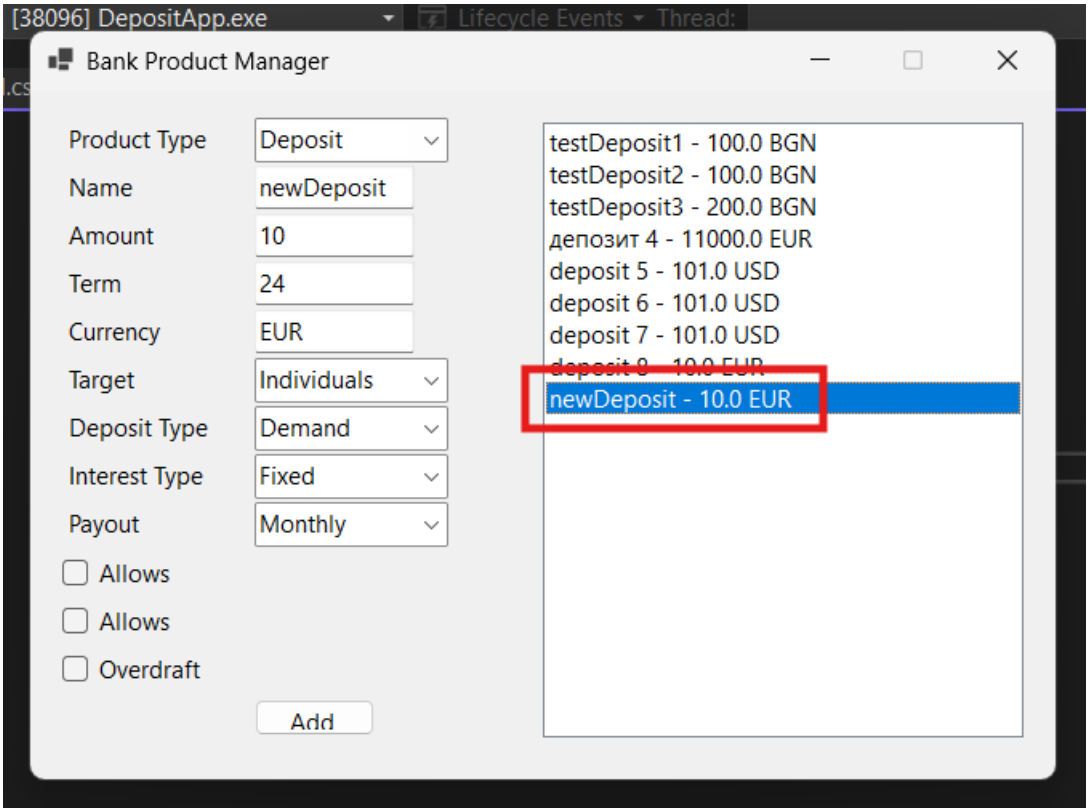
След условното разпределяне на тестовете по частите на системата, бяха изготвени две тестови сюити – една за десктоп частта и една за уеб сайтът. Всяка сюита съдържа различен брой тестове, като за всеки тест има идентификационен номер, статус, приоритет и трудност на бъговете, излезли от неговото изпълнение.

За реализацията на автоматичните тестове са използвани Pytest и Selenium с Python технологиите. Имплементиран е Page Object Model – разписани са базовите методи и локатори за цялата уеб част и се достъпват от отделна

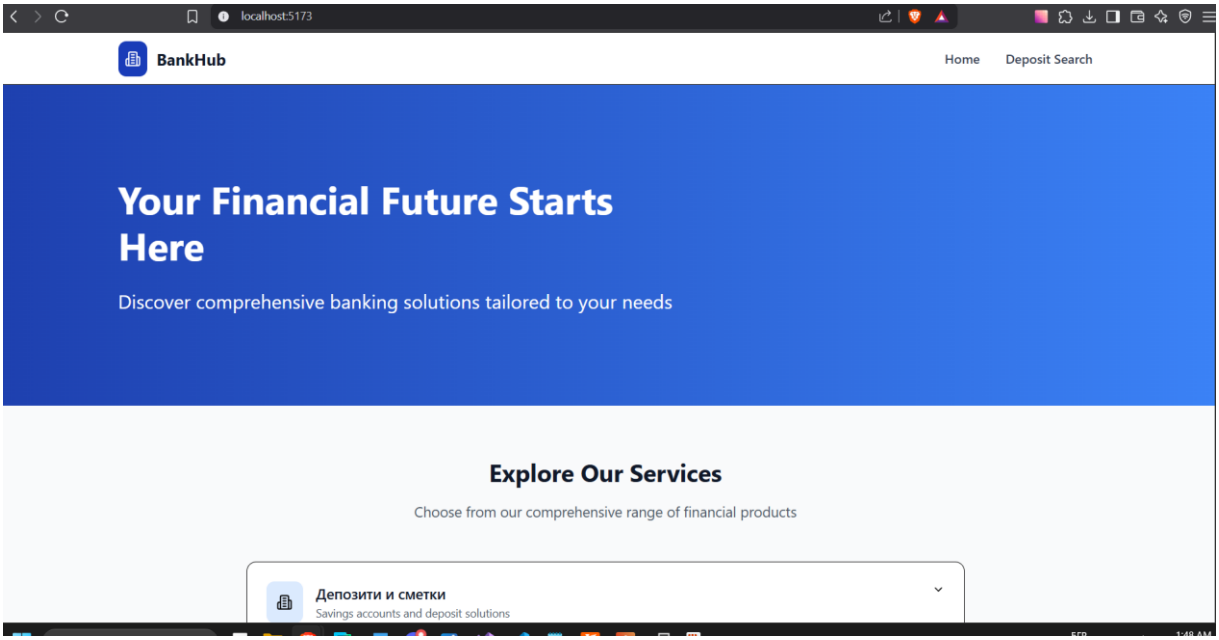
тестова директория. След изпъление на автоматичния скрип с командатат pytest, се генерира и справка за изпълнение тестове в html файл. Тестовете се изпълняват едновременно в два браузъра – Chrome и Firefox. Добавени за конфигурационни файлове – config.py, conftest.py и pytest.ini, които правят изпълнението в два браузъра възможно, генериране на справка след изпълнение на скрипта и форматиране на резултата от тестовете в терминала.

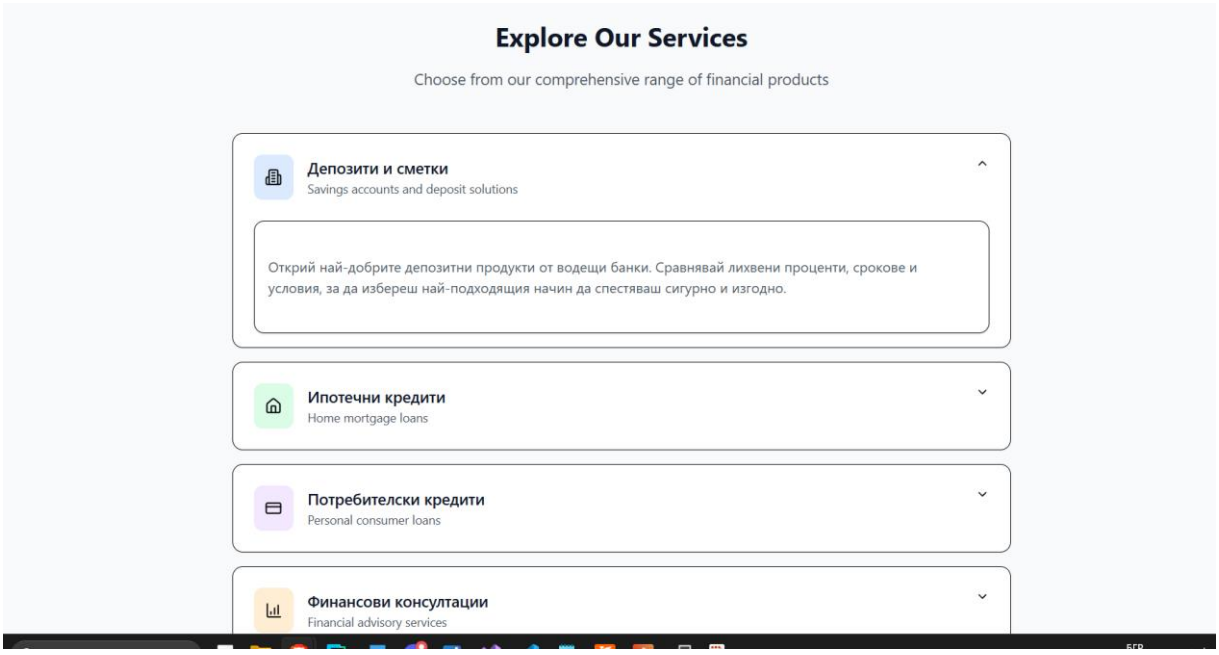
7. Практически пример за работа с приложението

За практическият пример, нека да започнем с десктоп приложението. Първото нещо, което е добре да направим, след като сме заредили BankAPI сървъра и сме стартирали DepositApp е да направим депозитен елемент. Например, нека да направим безсрочен депозит с име ‘newDeposit’’. Минималната му сума ще бъде 10 с валута EUR за срок от 24 месеца. Ще таргетира физически лица. Неговата лихва ще бъде фиксирана и няма да разрешава оувърдрафт.

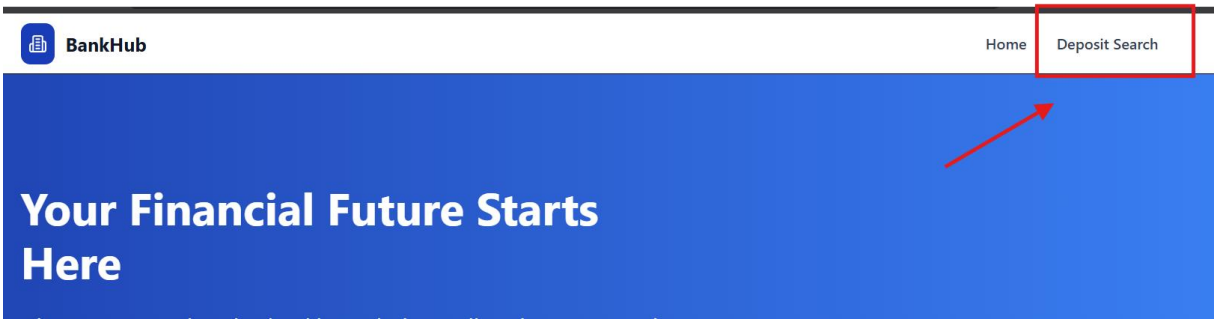


След това, нека да влезем в системата – за целта стартираме фронт енд проекта и отваряме съответния линк, генериран в терминала. Първото, което виждаме е началната страница:

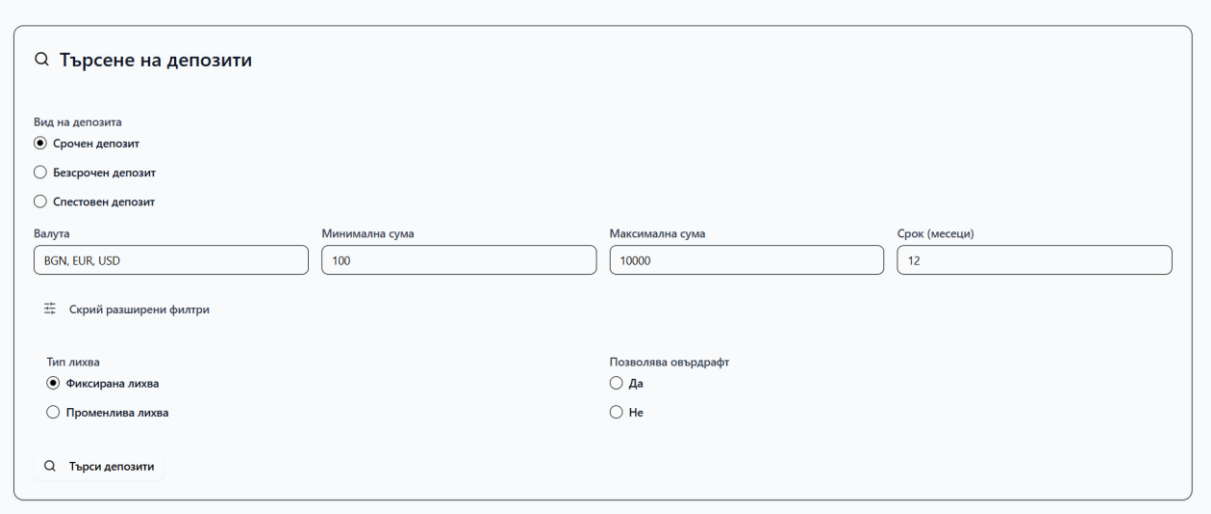




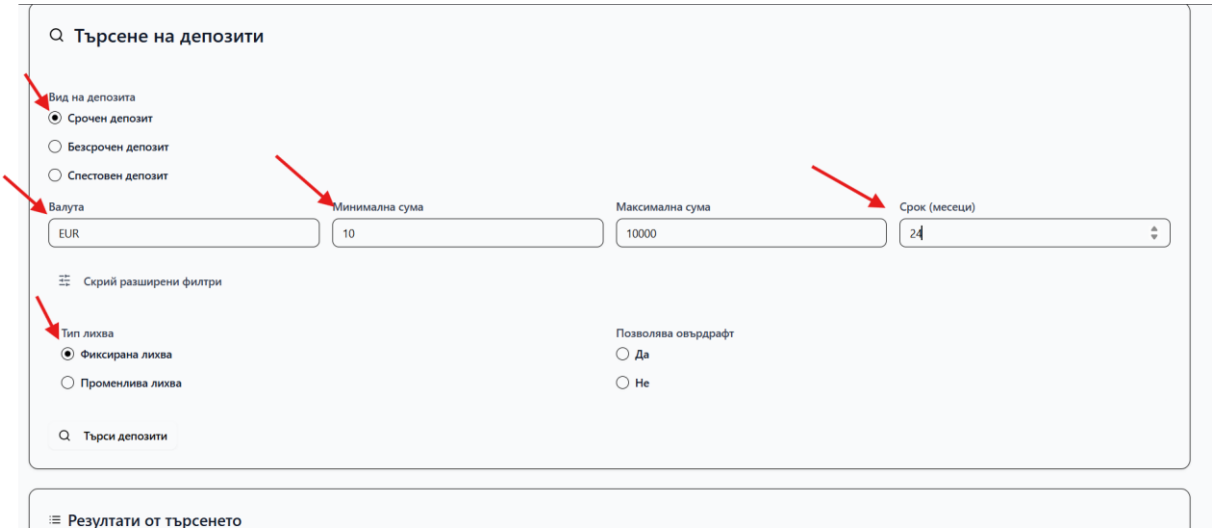
От нея можем да отидем в страницата за намиране на депозит -



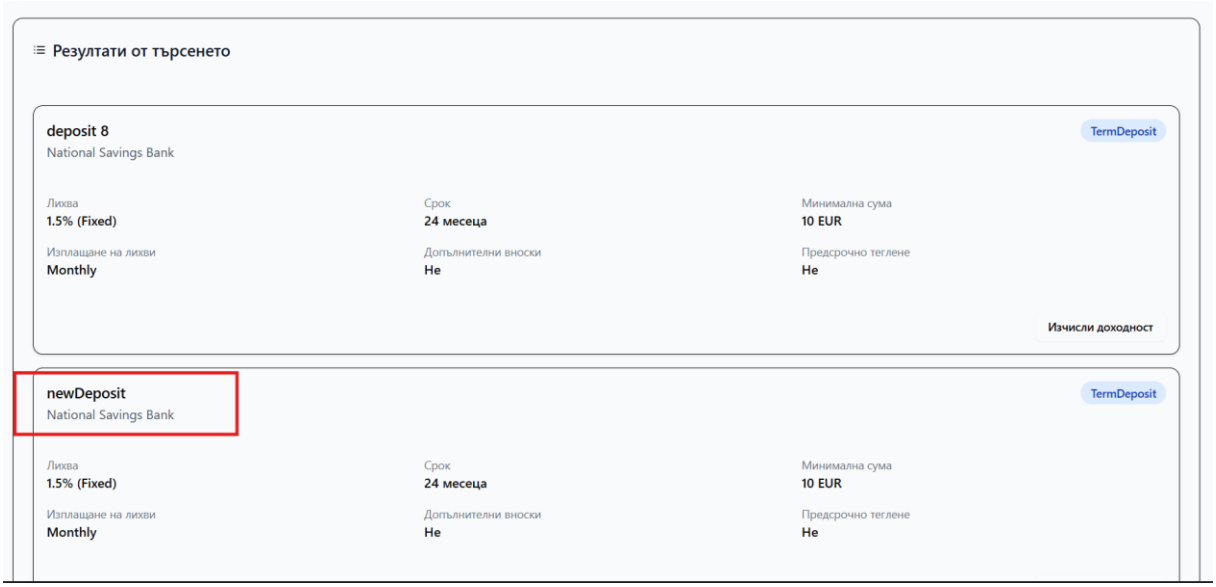
Там вече виждаме предоставените ни филтри от системата:



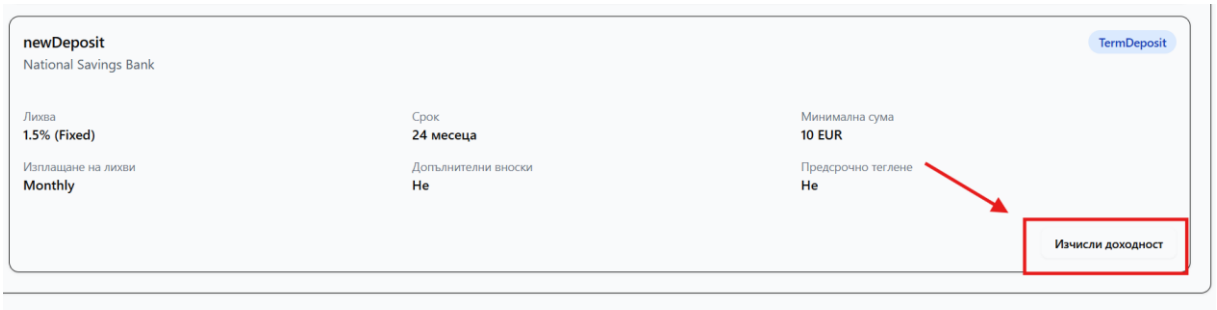
Нека да въведем някои от данните, с които е създаден newDeposit:



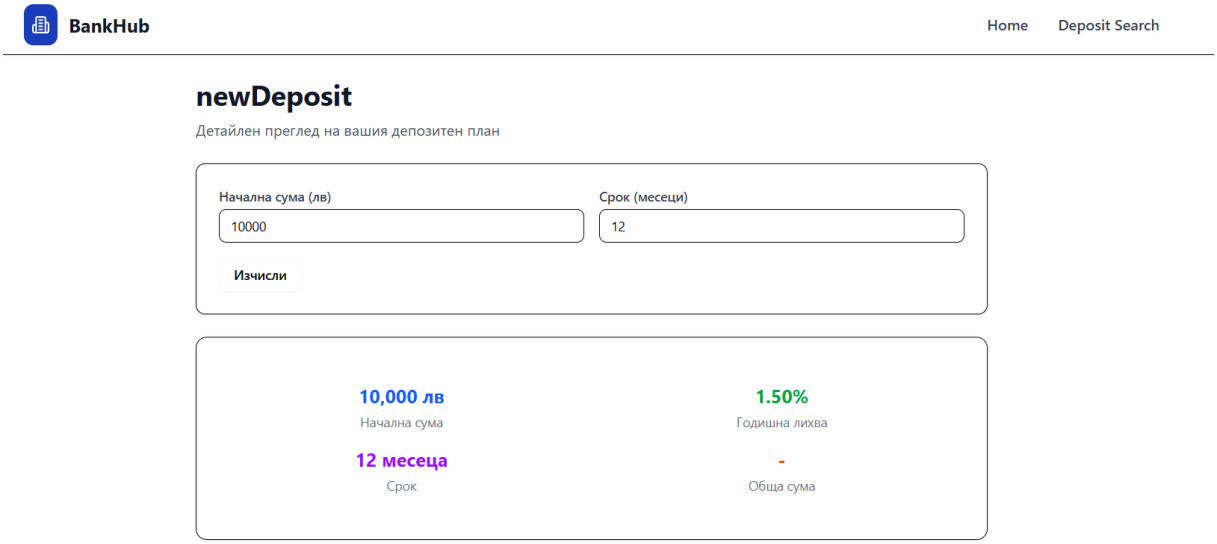
След натискане на бутона за изпълнение на търсенето, страницата ни визуализира желаният депозит:



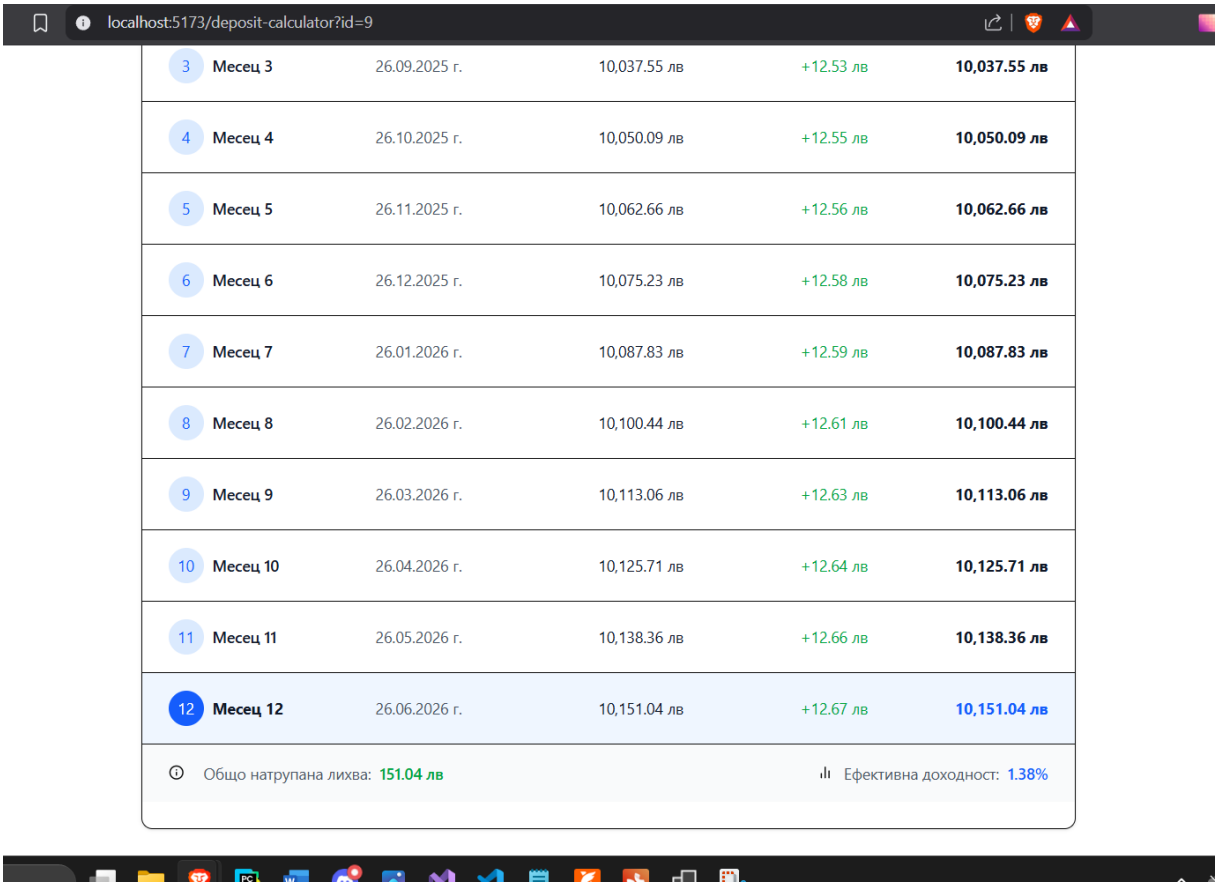
От него можем да отидем да разгледаме разплащателния план към конкретната оферта с бутона „Изчисли доходност“:



Навигирани сме към страницата за изчисление на разплащателния план. Той ще се зареди само след попълване на начална сума и срока за депозита. Опция за валута не е добавена, тъй като депозитната оферта е избрана с точно определена валута:



След въвеждане на данните, разплащателният план е генериран в следния вид:



Показва подробна информация за изплащането по време на периода по дати, суми, натрупана лихва и общо натрупана лихва за целия период, както и процент на ефективна доходност.