**ДИПЛОМЕН ПРОЕКТ**

**ЗА ДЪРЖАВЕН ЗРЕЛОСТЕН ИЗПИТ**

по професия код 481030 „Приложен програмист“

специалност код 4810301 Приложно програмиране“

ТЕМА: „ПРИЛОЖЕНИЕ ЗА УПРАВЛЕНИЕ НА КНИГОХРАНИЛИЩЕ“

Автор:

Александър Иванов Баев, клас XII В

Ръководител:

Име, Фамилия

БургасСЪДЪРЖАНИЕ

[1 Увод 3](#_Toc155035766)

[2 Цели и обхват на софтуерното приложение 3](#_Toc155035767)

[3 Анализ на решението 4](#_Toc155035768)

[3.1 Потребителски изисквания и работен процес 4](#_Toc155035769)

[3.2 Примерен потребителски интерфейс 5](#_Toc155035770)

[3.3 Диаграми на анализа 5](#_Toc155035771)

[3.4 Модел на съдържанието / данните 5](#_Toc155035772)

[4 Дизайн 6](#_Toc155035773)

[4.1 Реализация на архитектурата на приложението 6](#_Toc155035774)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 6](#_Toc155035775)

[4.3 Организация и код на заявките към база от данни 6](#_Toc155035776)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 6](#_Toc155035777)

[5 Ефективност и бързодействие на решението 7](#_Toc155035778)

[6 Тестване 7](#_Toc155035779)

[7 Заключение и възможно бъдещо развитие 7](#_Toc155035780)

[8 Използвани литературни източници и Уеб сайтове 7](#_Toc155035781)

[9 Приложения 8](#_Toc155035782)

[10 Критерии и показатели за оценяване 9](#_Toc155035783)

# Увод

В днешния дигитален свят ефективното управление на книжните ресурси и библиотечните системи е от съществено значение. Традиционните методи за проследяване и управление на инвентара на книги често са тромави, изискват ръчен труд и са податливи на грешки. Това затруднява работата на администраторите и ограничава достъпа на читателите до необходимите ресурси. Освен това, липсата на автоматизирана система за управление на книжния фонд води до затруднения при организирането на наличните ресурси, следенето на наемите и поддръжката на актуална база данни.

Настоящият проект има за цел да създаде уеб базирано приложение за управление на книгохранилище, което автоматизира и улеснява процесите по управление на инвентара, отдаване под наем на книги и следене на наличността им. Чрез внедряването на модерни технологии се осигурява по-добра организация, подобрена ефективност и по-лесен достъп до книжните ресурси. Системата позволява на администраторите да регистрират нови книги, да управляват информацията за тях, както и да проследяват кои книги са налични и кои са в процес на наемане. Освен това, приложението предоставя възможност за генериране на отчети, което подпомага анализирането на използваемостта на ресурсите и улеснява вземането на управленски решения.

Важна част от разработката на този проект е използването на подходящи софтуерни технологии и архитектури, които гарантират стабилност, мащабируемост и лесна поддръжка на системата. За моделирането на решението се използва езикът UML, като различните диаграми (диаграма на случаите на употреба, диаграма на класовете и диаграма на дейностите) се създават в средата за моделиране Lucidchart. Тези диаграми илюстрират логическата структура на системата, взаимодействието между потребителите и различните компоненти на приложението, както и основните процеси, които системата поддържа.

Проектът се основава на многослойна архитектура, което осигурява ясно разделение между различните компоненти на приложението – база данни, бизнес логика и потребителски интерфейс. В технологичен аспект се използва ASP.NET Core за сървърната част, Entity Framework Core за работа с базата данни и MVC/Razor Pages за фронтенд слоя. Това позволява създаването на гъвкаво и мащабируемо решение, което може да се разширява в бъдеще с нови функционалности и интеграции.

Останалата част от документацията е структурирана, както следва:

* **Цели и обхват на софтуерното приложение**
  + Описва целите и обхвата на приложението, като дефинира основните функционалности, които се реализират.
* **Анализ на решението**
  + **П**редставя анализа на решението, включително потребителските изисквания, работния процес и логическата структура на данните.
* **Дизайн**
  + Разглежда архитектурата и дизайна на приложението, като обяснява как различните компоненти са организирани и как взаимодействат помежду си.
* **Ефективност и бързодействие на решението**
  + **А**нализира ефективността и производителността на решението, като разглежда начини за оптимизация и подобрение на работата на системата.
* **Тестване**
  + **Точката** е посветена на тестването на системата, включително стратегия за извършване на Unit тестове и интеграционни тестове.
* **Заключение и възможно бъдещо развитие**
  + Разглежда възможности за бъдещо развитие и разширяване на функционалността на приложението.
* **Използвани литературни източници и Уеб сайтове**
  + Съдържа използваните литературни източници и уеб сайтове, които служат като основа за изследването и разработката.
* **Приложения**
  + Включва приложения с допълнителни материали, като детайлни диаграми и примери за код.

# Цели и обхват на софтуерното приложение

Софтуерното приложение за управление на книгохранилище е създадено с цел да автоматизира процесите, свързани с управлението на книги, наематели и наемите на книги. Чрез интеграцията на съвременни технологии, приложението предоставя ефективен начин за администриране на библиотечния инвентар, управление на потребителите и генериране на отчети.

**Обхват на софтуерното приложение**

Приложението е предназначено за администратори, които да управляват книжните ресурси и процесите, свързани с тях. Те имат достъп до следните функционалности:

* **Автентикация и сигурност**
  + Вход в системата (login)
  + Възстановяване на забравена парола (forgot password)
  + Нулиране на парола (reset password)
  + Промяна на паролата от профила на администратора (change password)
* **Управление на книги**
  + Добавяне на нови книги в системата
  + Редактиране на информация за вече въведени книги
  + Преглед на детайлите на дадена книга
  + Изтриване на книги от каталога
* **Управление на наематели**
  + Добавяне на нови наематели
  + Поддържане на информация за съществуващи наематели
  + Връзка с потребителите
* **Управление на наеми**
  + Добавяне на нови наеми
  + Проследяване на статуса на даден наем (активен, просрочен, върнат)
* **Генериране на отчети**
  + Генериране на PDF подробен месечен отчет за наемите
* **Статистика**
  + Преглед на броя налични книги спрямо общия брой
  + Най-често наеманите книги
  + Най-активните наематели
  + Преглед на завършените наеми спрямо общия брой
* **Диаграма**
  + Диаграма, показваща месечния брой наеми

Приложението осигурява интуитивен потребителски интерфейс, който улеснява достъпа до информация и минимизира времето, необходимо за изпълнение на административни задачи.

**Цели на софтуерното приложение**

Основната цел на разработката е да създаде ефективно, сигурно и лесно за използване уеб приложение, което да подпомага управлението на книгохранилището. Това се постига чрез:

* Автоматизиране на ръчните процеси по администриране на книжните ресурси и наемите
* Подобряване на контрола върху наличността на книгите
* Улесняване на администраторите при поддръжката на базата с наематели
* Осигуряване на механизъм за генериране на полезни отчети и анализ на данните
* Гарантиране на сигурност при достъпа до системата чрез надеждни механизми за автентикация и управление на потребителски профили

Разработеното приложение оптимизира работата на книгохранилищата, намалява риска от грешки и повишава ефективността на администраторите при управлението на библиотечните ресурси.

# Анализ на решението

## Потребителски изисквания и работен процес

Настоящото софтуерно приложение е предназначено за управление на книгохранилище, като основен негов потребител е администраторът. Той има достъп до всички функционалности, включително управление на книги, наематели и наеми, както и генериране на отчети. За да осигури надеждна и ефективна работа, системата следва добре дефиниран работен процес, който включва три основни етапа: вход**,** обработкаиизход.

**Входни данни и тяхното придобиване**

Входните данни в системата са от съществено значение за нейната функционалност и се въвеждат от администратора чрез потребителския интерфейс. Те включват:

* **Данни за автентикация и управление на достъпа**

За да използва системата, администраторът трябва да се идентифицира чрез форма за вход. Тази функционалност включва:

* Въвеждане на имейл и парола.
* Проверка за коректност на въведените данни.
* Възможност за нулиране на паролата.
* Възможност за смяна на паролата в случай на необходимост.

Тези процеси са обезпечени чрез ASP.NET Identity, което осигурява сигурно управление на потребителски акаунти.

* **Данни за книги**

За да бъде една книга част от системата, тя трябва да бъде регистрирана чрез формуляр, включващ:

* Заглавие
* Автор
* Жанр
* Място на книгата
* Информация
* Дата на публикуване
* Количество
* ISBN
* SKU
* Брой страници
* Възрастов диапазон
* Корица на книгата
* Вид корица

При добавяне на книга се извършва валидация, като системата проверява дали полетата са попълнени коректно и дали корицата отговаря на зададените изисквания за файлов формат (.jpeg, .png или .svg).

* **Данни за наематели**

За да може дадено лице да наема книги, то трябва да бъде добавено като наемател. Администраторът въвежда следната информация:

* Име
* Имейл
* Телефонен номер
* **Данни за наеми**

Когато администраторът създава нов наем, той избира:

* Книга, която ще бъде наета.
* Наемател, който взема книгата.
* Начална дата на наема.
* Краен срок за връщане.

При добавяне на наем, статусът на наемът се променя на „**Потвърден**“. Когато наемът започне се променя на „**Активен**“. Когато текущата дата е по-късна от крайния срок за връщане и книгата не е върната, статусът става „**Просрочен**“. След връщане се отбелязва като „**Завършен**“.

**Обработка и съхранение на данните**

След като потребителят въведе данните, те преминават през няколко основни етапа на обработка:

* **Валидация на входните данни**

Системата проверява дали въведената информация е коректна и пълна. Тази проверка включва:

* **Данни за книги** – Не се допуска добавяне на дублиращи се книги със същото заглавие и автор.
* **Данни за наематели** – Имейлът трябва да бъде валиден.
* **Данни за наеми** – Системата не позволява наемане на книга, която вече не е в наличност.
* **Запазване на информацията в база данни**

Всички данни се съхраняват в релационна база данни чрез **Entity Framework Core**. Връзките между обектите са дефинирани по следния начин:

* Всяка книга може да бъде наета многократно.
* Всеки наемател може да има няколко активни наема.
* Всеки наем е свързан с точно една книга и един наемател.
* **Обновяване на състоянието**
* При добавяне на наем - статусът на наемът се променя на „**Потвърден**“.
* При започнат наем - статусът се променя на „**Активен**“.
* Когато текущата дата е по-късна от крайния срок за връщане и книгата не е върната - статусът се променя на „**Просрочен**“.
* При върнат наем – статусът се променя на „**Завършен**“.

**Изход и използване на информацията**

След като данните са обработени и съхранени, приложението генерира изходна информация, която се предоставя на администратора под различни форми:

* **Таблици с книги, наематели и наеми**
* Администраторът може да преглежда списъците, както и да филтрира и търси по различни критерии.
* Таблицата за книгите е сортирана по азбучен ред спрямо заглавието. Има филтърната опция „търсачка“, като може да се търси по заглавие, автор, жанр или място.
* Таблицата за наематели е сортирана по азбучен ред спрямо имената на наемателите. Има филтърната опция „търсачка“, като може да се търси по име, имейл или телефонен номер.
* Таблицата за наемите е сортирана по дата на наем и популярност. Има филтърната опция „търсачка“, като може да се търси по статус, име на книгата или наемател.
* **Генериране на отчети**

В началната страница администраторите могат да генерират PDF месечен отчет за наемите.

* **Статистика**

В началната страница администраторите могат видят различни справки като:

* Преглед на броя налични книги спрямо общия брой
* Най-често наеманите книги
* Най-активните наематели
* Преглед на върнатите наеми спрямо общия брой
* **Диаграма**

В началната страница администраторите могат видят диаграма, показваща месечния брой наеми.

**UML диаграми**

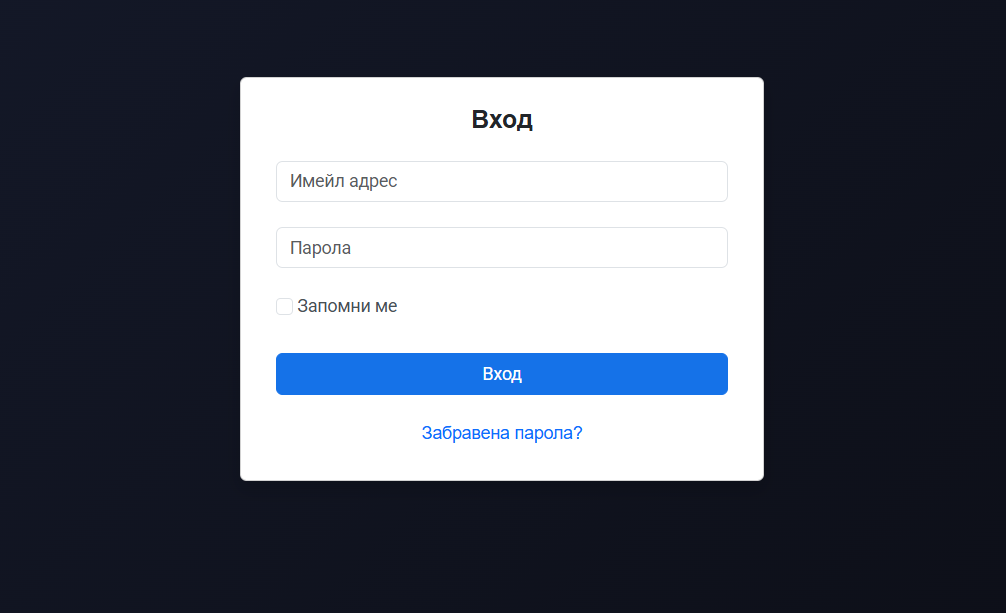
За онагледяване на работния процес са създадени **UML диаграми**, включително:

* **Диаграми на случаи на употреба** – Описват основните взаимодействия между администратора и системата.
* **Диаграми на дейностите** – Представят последователността на действията при добавяне на книга, създаване на наем и управление на системата.

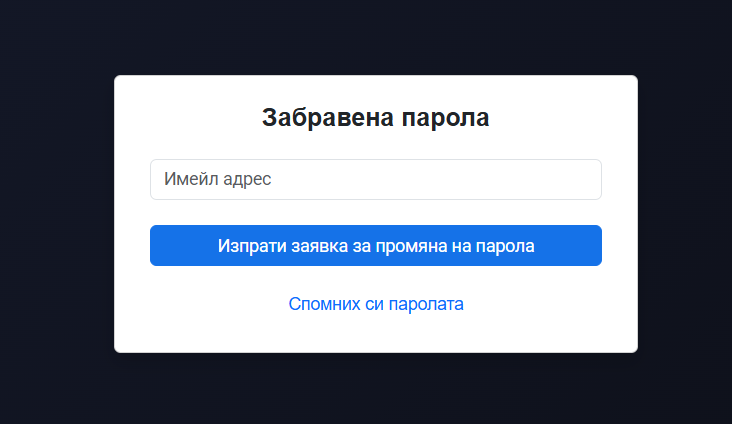
Всички диаграми са създадени в **Lucidchart**, като оригиналните им файлове са **добавени в репозиторито**. Те спазват стандарта **UML**.

## Примерен потребителски интерфейс

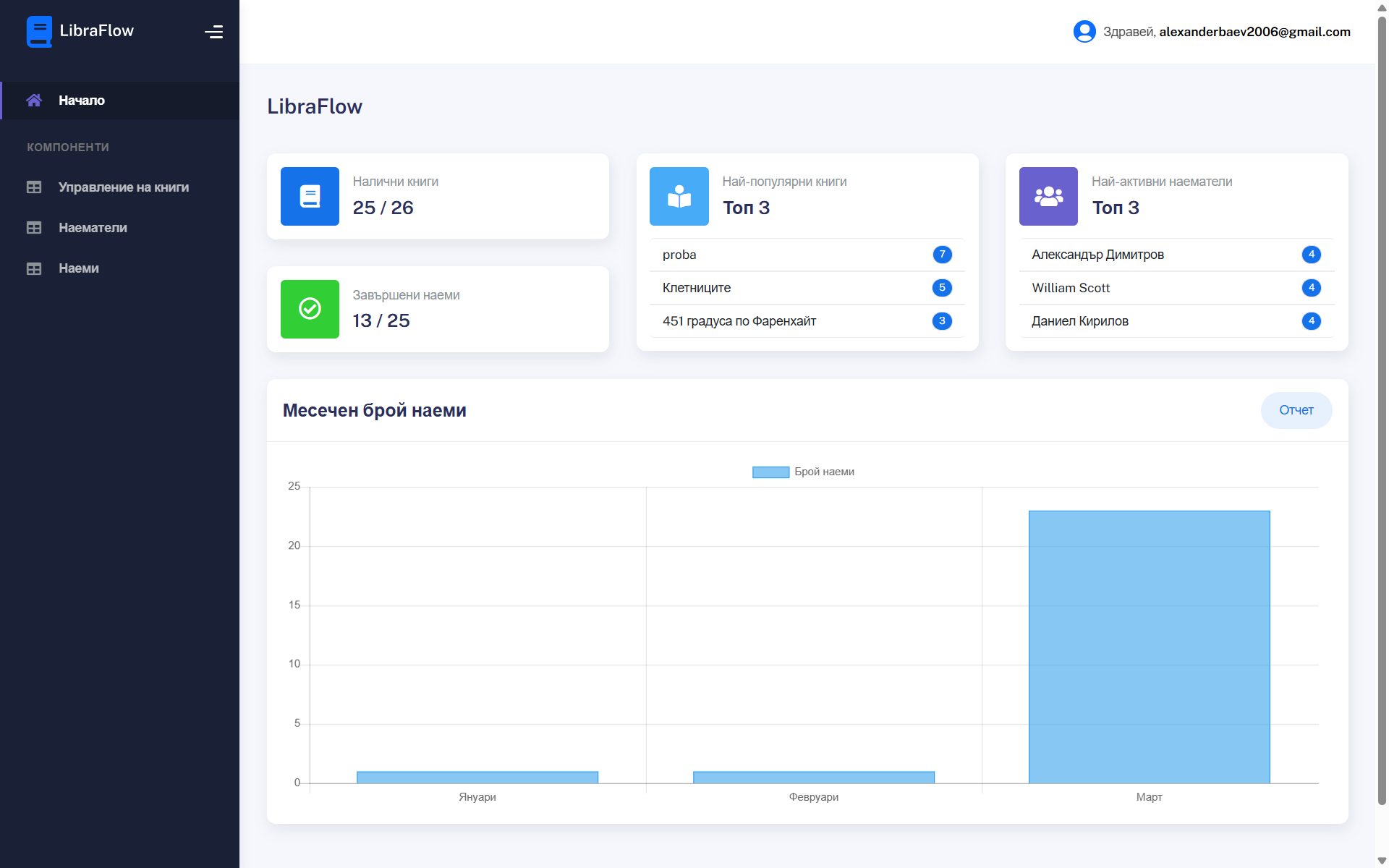
Графичният потребителски интерфейс (GUI) на уеб приложението е проектиран с цел улеснена навигация и интуитивно управление на книжния инвентар, наемателите и наемите. Визуалното оформление е съобразено с модерните уеб стандарти и използва чист и опростен дизайн, който позволява бърз достъп до основните функции.



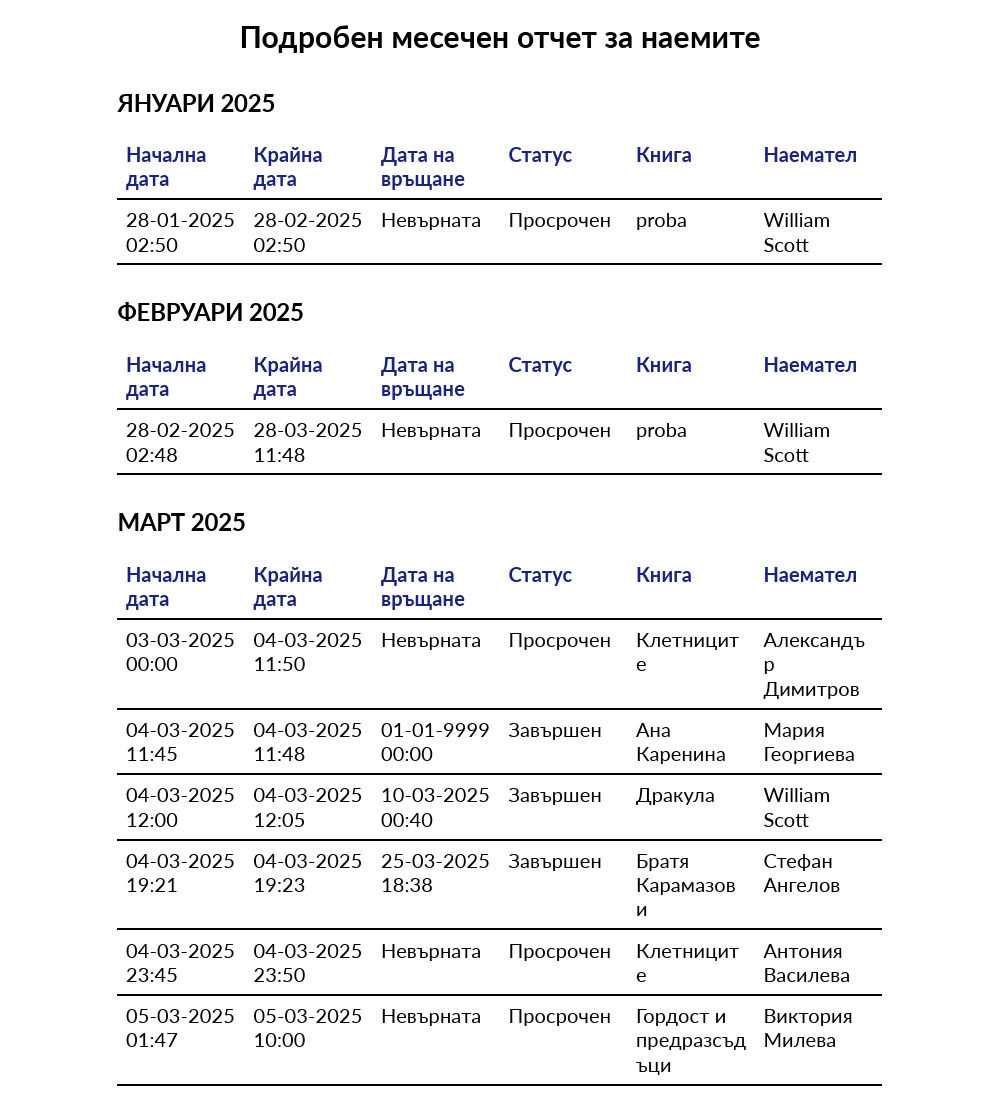
Вход



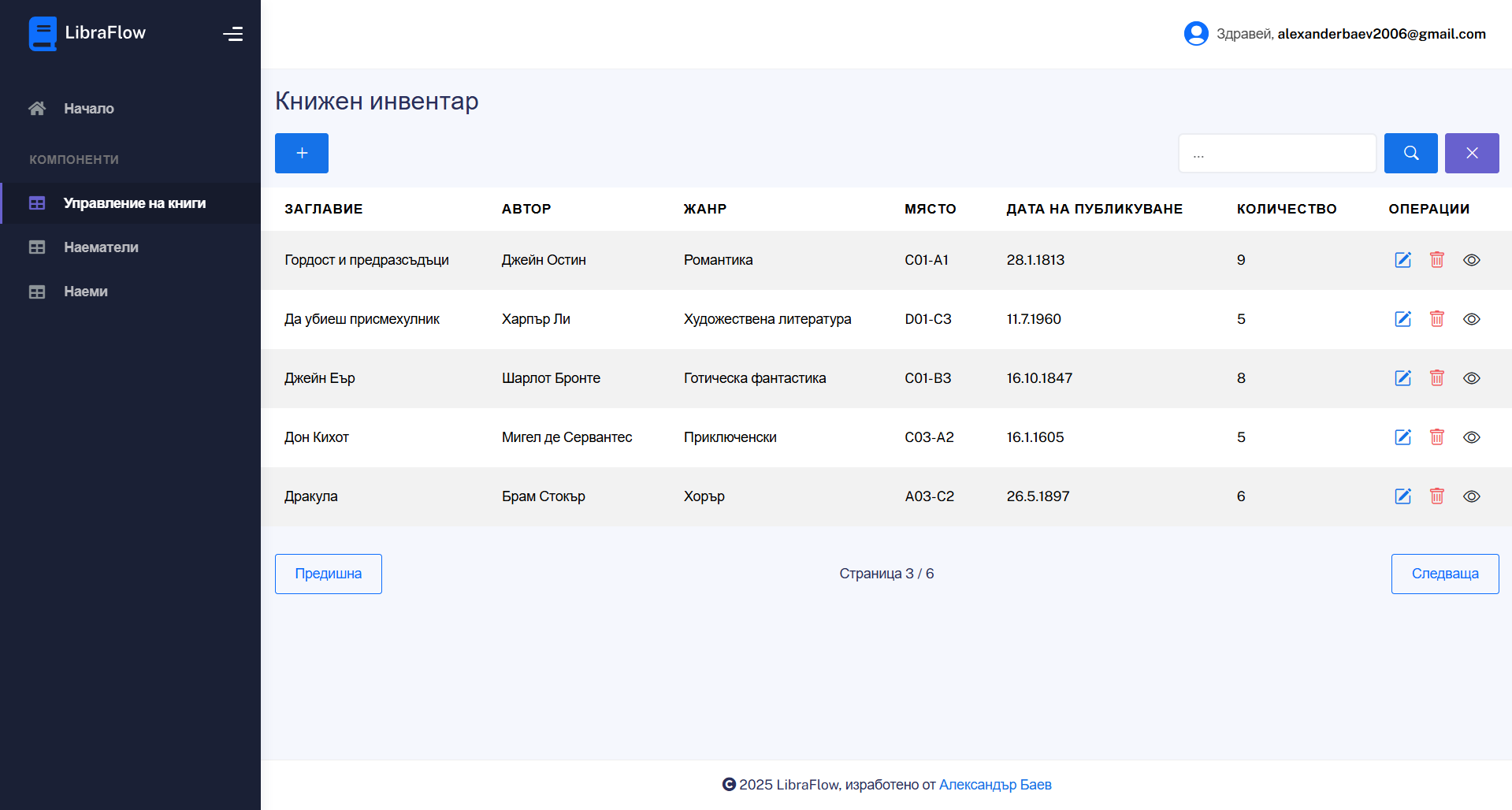
Забравена парола



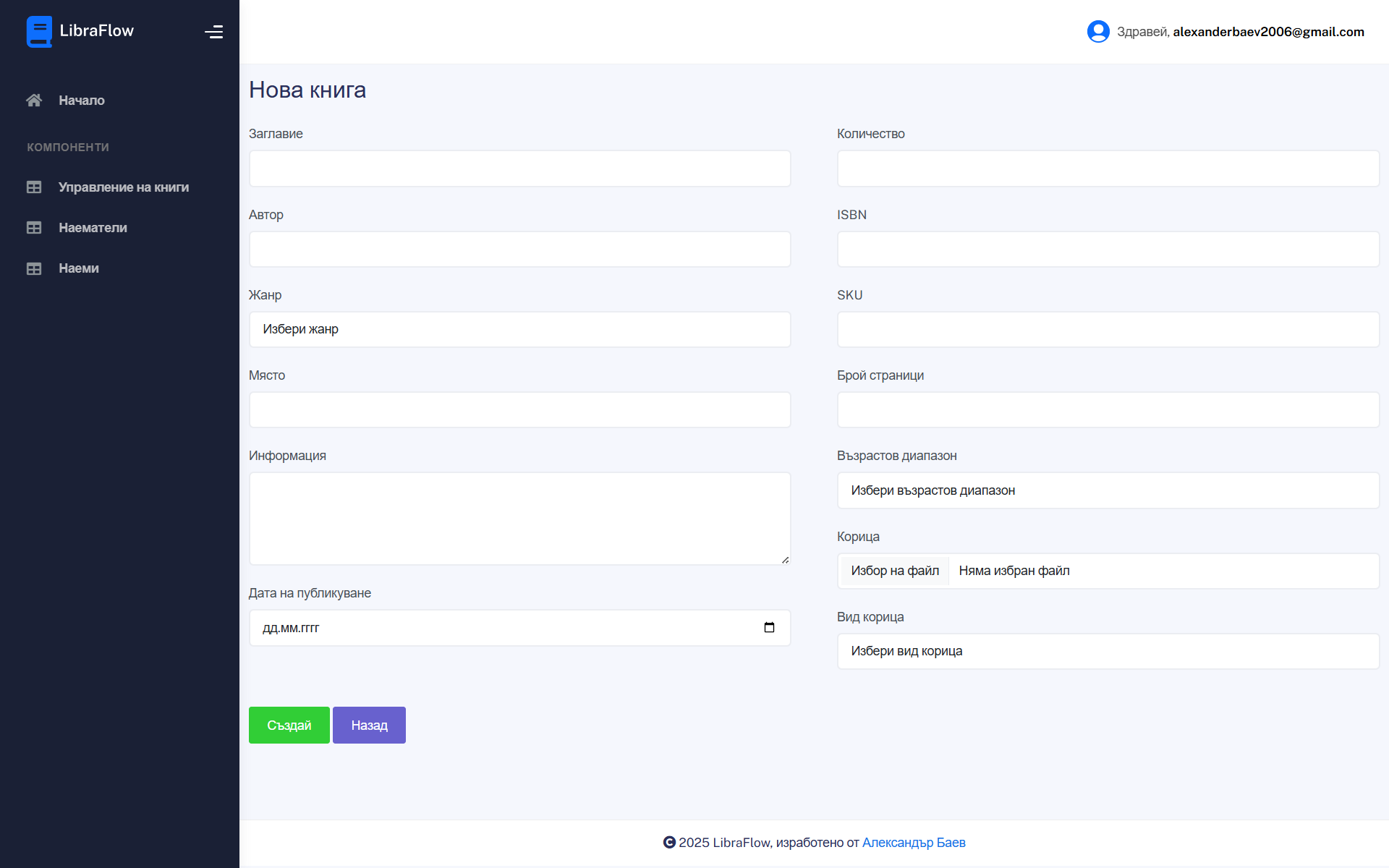
Начална страница



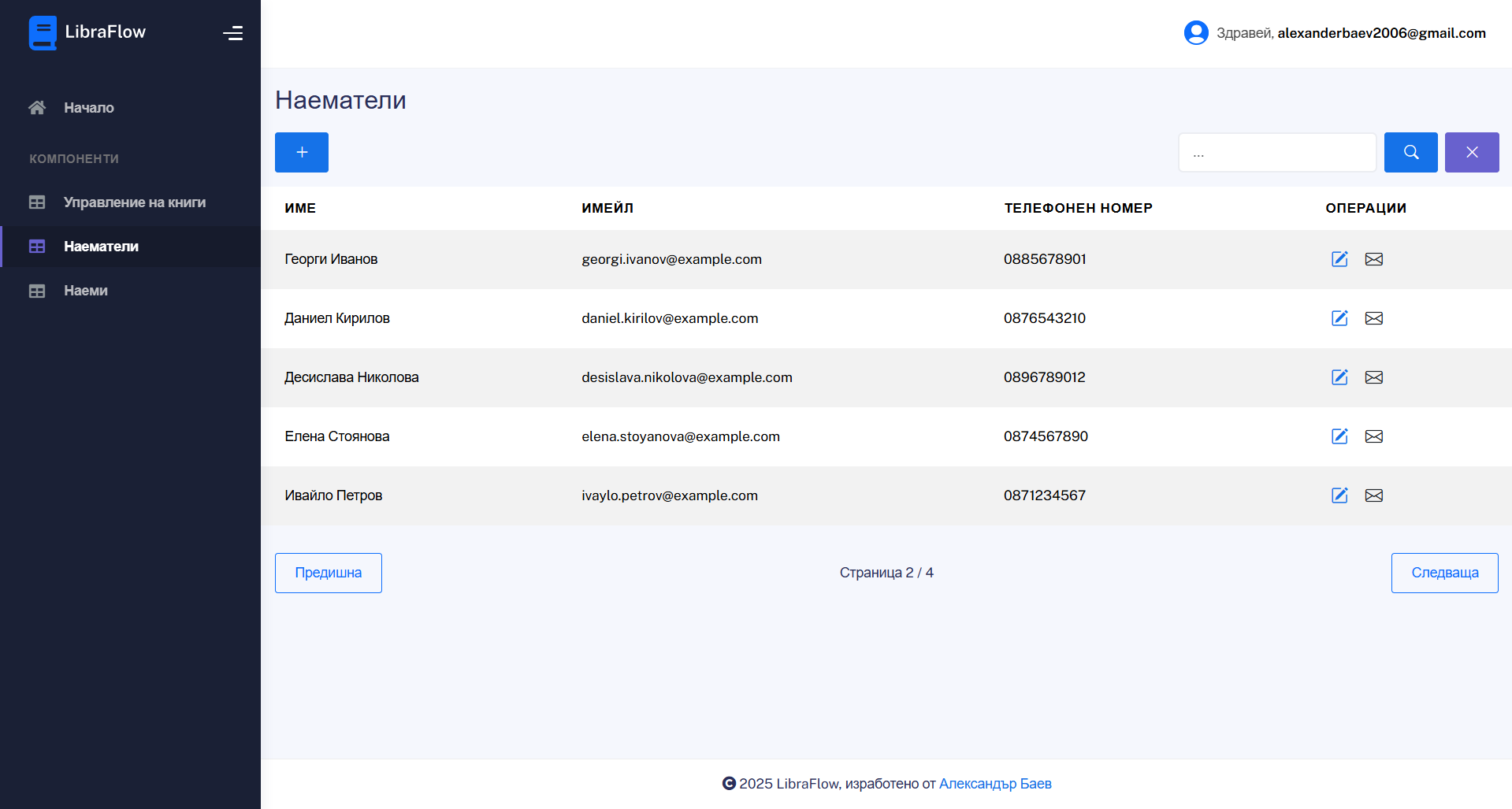
Генериране на отчети - Експортира справка във формат PDF



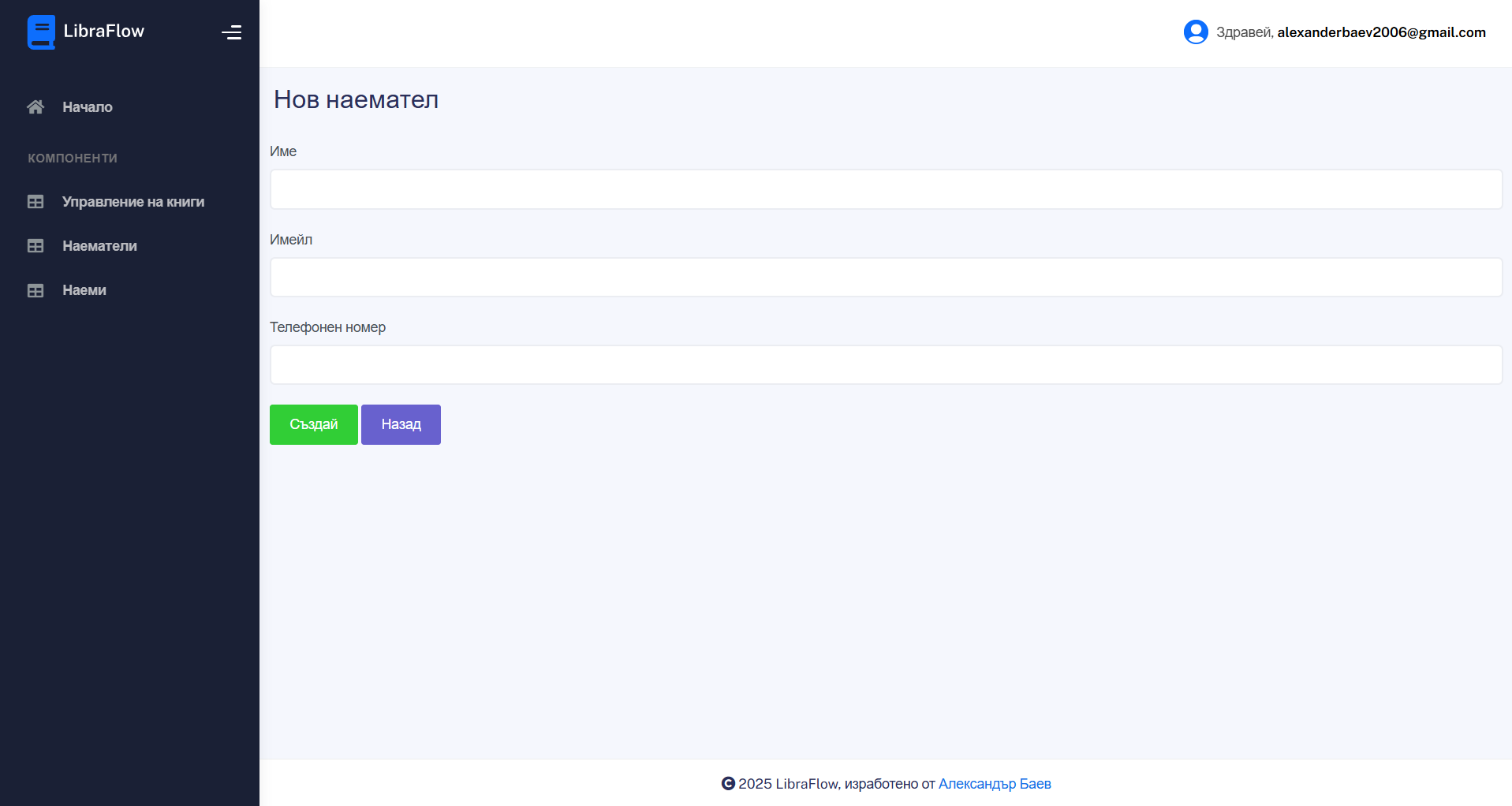
Книжен инвентар



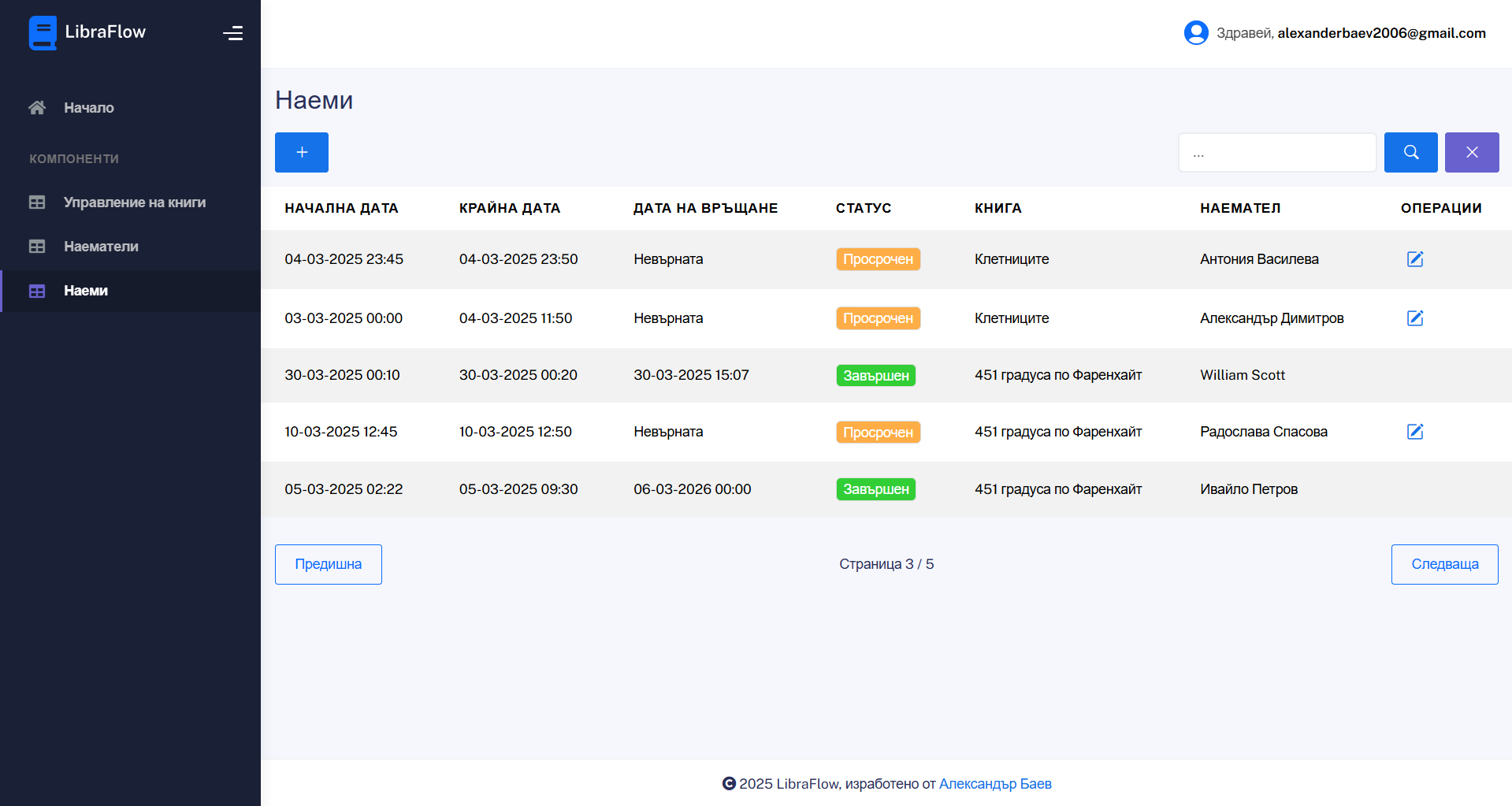
Добавяне на нова книга



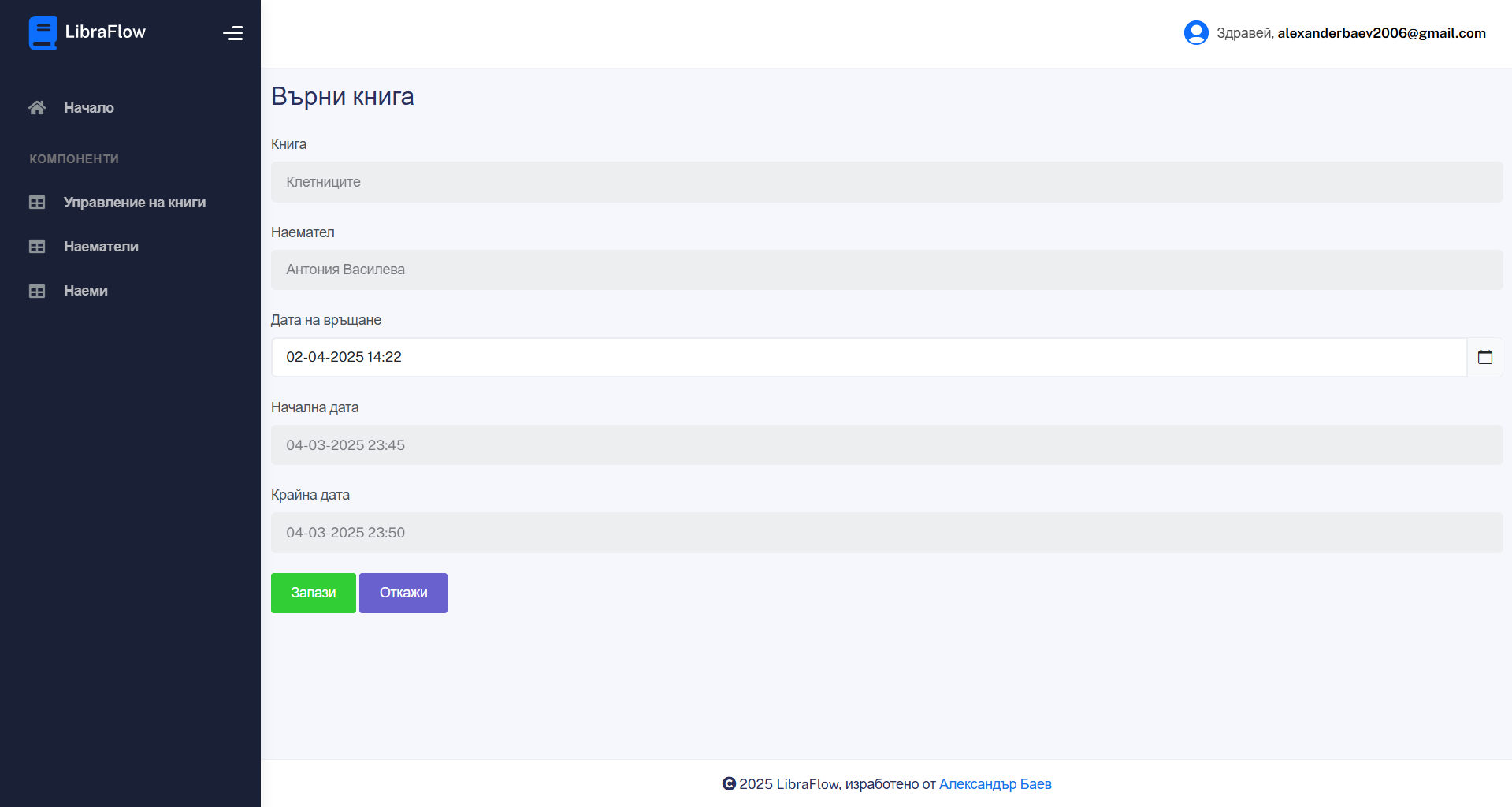
Наематели



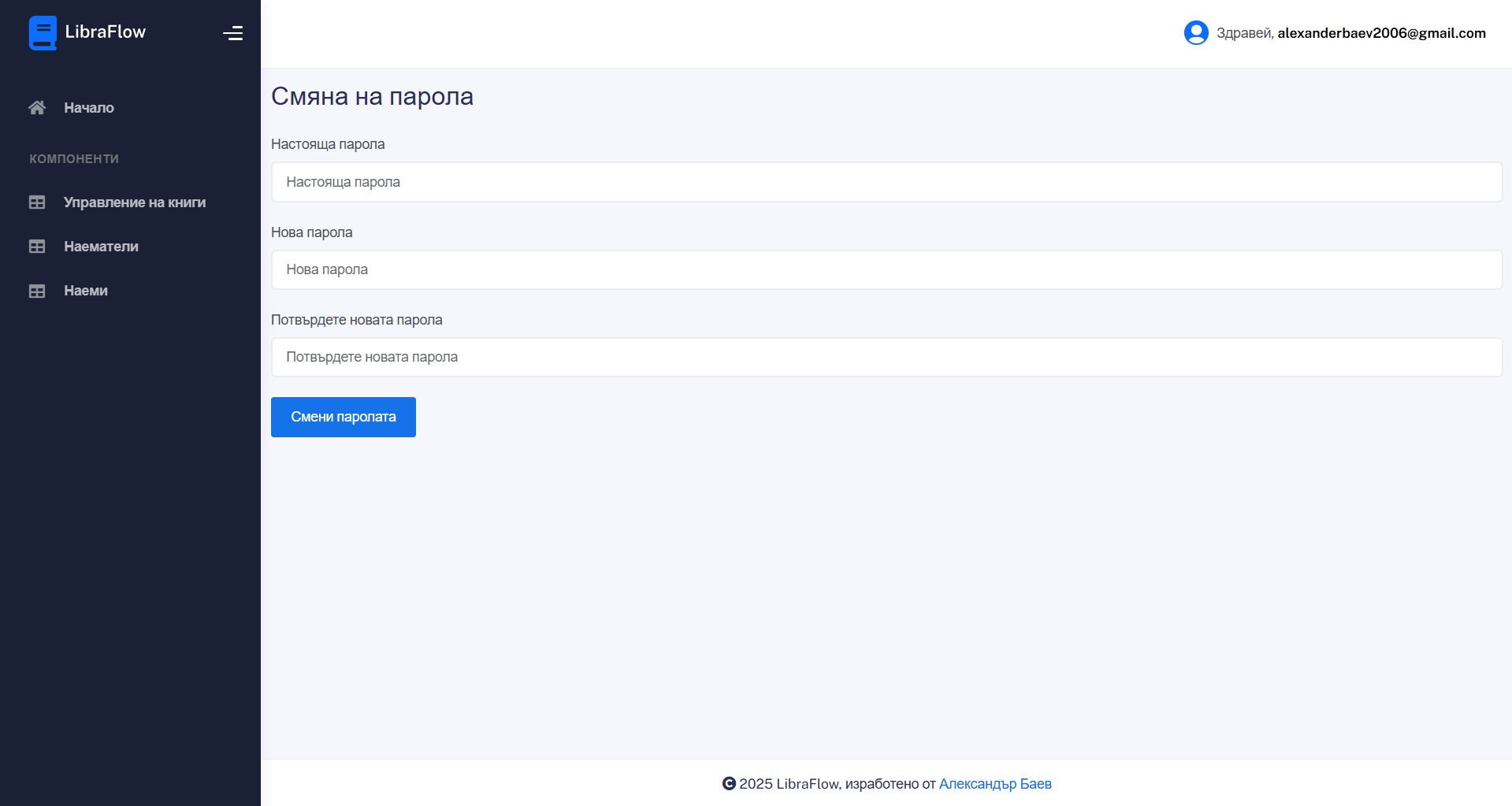
Добавяне на нов наемател



Наеми



Връщане на книга



Смяна на парола

## Диаграми на анализа

В този раздел са представени UML диаграми, които илюстрират архитектурата и логиката на разработваното приложение. Те помагат за по-доброто разбиране на връзките между отделните компоненти, потока на данни и процесите, които протичат в системата.

## Модел на съдържанието / данните

Тук опишете модела на данните/съдържанието - текстово, графично и евентуално аудио/видео съдържание), което ще представите в проекта си. Опишете размера и типа на данните/файловете и начина на кодиране за всеки от ресурсите.

Ако ползвате в проекта си текстово и/или мултимедийно съдържание от различни типове, представете неговата структура, напр. посредством таксономия, типология, онтология или други схеми за представяне на структурата от категории, под-категории, типове и т.н., както и техните взаимовръзки с други категории или типове, напр. географски региони и дялове, области/сегменти от промишлеността, и др.

След структурата опишете и възможните характеристики, атрибути и честота на срещане на всеки един ресурс в съдържанието (категория, тип, екземпляр, връзка/релация и т.н.).

# Дизайн

Тази секция представя дизайна на решението на проблема за проекта ви. Опишете каква софтуерна платформа сте избрали за вашето решение /напр. .NET, java/. Представете схема на софтуерната архитектура на решението /по модули и/или слоеве/ с диаграма на разгръщането, както и диаграми на класовете на дизайна /с ограничения, описани на OCL/, диаграми на времето /за задаване на времена за синхронизация и комуникация в решението/ и компонентни диаграми. Илюстрирайте решението с извадки от генериран сорс код.

## Реализация на архитектурата на приложението

## Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой.

## Организация и код на заявките към база от данни

Описание на инструментариума за достъп до базата данни от гледна точка на програмния код. Описание на методите за извличане, добавяне и изтриване на обекти в базата данни.

## Наличие на потребителски интерфейс (конзолен, графичен, уеб)

Описание на основните функционалности на интерфейса на приложението.

Забележка: Няма формално изискване на определен брой диаграми от даден вид, за даден брой проектанти.

# Ефективност и бързодействие на решението

Съдържа описание и анализ на известните решения, като се цитират съответните литературни източници.

# Тестване

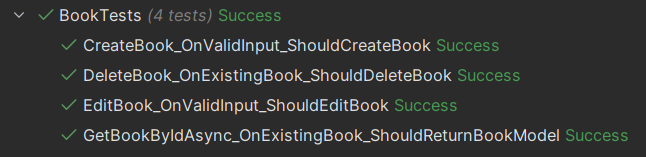
В този раздел са описани тестовите случаи и видовете тестване, които са предвидени за реалното изпълнение на проекта. Тестването включва:

**Видове тестове**

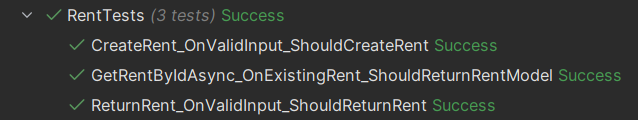
* **Unit тестове** – покриват основните функционалности на системата, като управление на книгохранилищните инвентари и наемите.
* **Интеграционни тестове** – проверяват взаимодействието между различните модули, включително генериране на отчети.

**Unit тестове**

* **Тестове за управление на книгохранилищни инвентари**



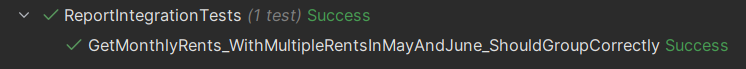
* Тест за извличане на книга по ID – проверява дали системата успешно връща информация за съществуваща книга.
* Тест за създаване на книга – проверява дали книга може да бъде създадена успешно и данните ѝ да се запазят в базата данни.
* Тест за редактиране на книга – тества дали промените в съществуваща книга се отразяват правилно.
* Тест за изтриване на книга – проверява дали книга се премахва успешно от системата.
* **Тестове за управление на наемите**



* Тест за извличане на наем по ID – проверява дали информацията за съществуващ наем се връща коректно.
* Тест за създаване на наем – гарантира, че наемът се създава и записва правилно.
* Тест за връщане на наем – проверява дали книгата правилно се отбелязва като върната и количеството в наличност се актуализира.

**Интеграционни тестове**

* **Тестове за генериране на отчети**



* Тест за извличане на месечни наеми – проверява дали отчетът за месечните наеми коректно групира данните и съдържа правилната информация.

**Среда за тестване**

* **База данни:** SQL Server (локална и тестова среда).
* **Рамки за тестване:** xUnit, FluentAssertions, Microsoft.AspNetCore.Mvc.Testing, Microsoft.AspNetCore.Hosting, Microsoft.EntityFrameworkCore.InMemory.
* **Инструменти за отчитане на тестовете:** Jetbrains Rider Test Runner.

# Заключение и възможно бъдещо развитие

В заключение, обобщете резултатите от работата ви по проекта, както и предимствата и ограничеността на използваните технологии / езици / методи. Укажете какви алтернативи могат да се използват и техните предимства и недостатъци. Опишете каква е използваемостта на подобни решения в практиката и какво бихте предложили като насоки за бъдещо развитие на вашето решение.

# Използвани литературни източници и Уеб сайтове

1. Официална документация на StyleCop Analyzers:
   1. https://github.com/DotNetAnalyzers/StyleCopAnalyzers
2. Unit тестване в .NET чрез xUnit
   1. https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-xunit
3. Microsoft.AspNetCore.Identity.EntityFrameworkCore
   1. <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-9.0&tabs=visual-studio>
4. QuestPDF
   1. <https://www.questpdf.com/quick-start.html>
5. SendGrid
   1. <https://docs.sendgrid.com/>
6. FluentAssertions
   1. <https://fluentassertions.com/introduction.html>
7. Martin, R. C. (2008). "Clean Code: A Handbook of Agile Software Craftsmanship". Prentice Hall.
8. Hunt, A., & Thomas, D. (1999). "The Pragmatic Programmer: Your Journey To Mastery". Addison-Wesley.
9. Watson, B. (2014). "Writing High-Performance .NET Code". Ben Watson.
10. Feathers, M. (2004). "Working Effectively with Legacy Code". Prentice Hall.

# Приложения

При необходимост можете да добавите и допълнителни секции под формата на апендикси. Таблица с диаграми, таблици и графики

Забележка:

1. Документацията на проекта се предава само в електронен вид в MS Word, чрез качването на архив с документа и останалите файлове по проекта, в задание за предаване на проект, в канала на екипа в Teams.
2. Кода на проекта, базата данни и документацията трябва да са налични в репозитори в GitHub, което е копие на заданието генерирано в организацията.