**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:

Димитър Янакиев

БургасСЪДЪРЖАНИЕ

[1 Увод 3](#_Toc155035766)

[2 Цели и обхват на софтуерното приложение 4](#_Toc155035767)

[3 Анализ на решението 7](#_Toc155035768)

[3.1 Потребителски изисквания и работен процес 7](#_Toc155035769)

[3.2 Примерен потребителски интерфейс 11](#_Toc155035770)

[3.3 Диаграми на анализа 18](#_Toc155035771)

[3.4 Модел на съдържанието / данните 22](#_Toc155035772)

[4 Дизайн 26](#_Toc155035773)

[4.1 Реализация на архитектурата на приложението 28](#_Toc155035774)

[4.2 Описание на слоевете, предназначението им, библиотеки и методи включени в съответния слой. 29](#_Toc155035775)

[4.3 Организация и код на заявките към база от данни 33](#_Toc155035776)

[4.4 Наличие на потребителски интерфейс (конзолен, графичен, уеб) 33](#_Toc155035777)

[5 Ефективност и бързодействие на решението 35](#_Toc155035778)

[6 Тестване 36](#_Toc155035779)

[7 Заключение и възможно бъдещо развитие 38](#_Toc155035780)

[8 Използвани литературни източници и Уеб сайтове 39](#_Toc155035781)

[9 Приложения 40](#_Toc155035782)

# 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 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 главно 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**
  + Преглед на наемателите
  + Връзка с потребителите
* **Управление на наеми**
  + Добавяне на нови наеми
  + Връщане на наеми
  + Проследяване на статуса на даден наем (активен, просрочен, върнат)
* **Генериране на отчети**
  + Генериране на PDF подробен месечен отчет за наемите
* **Статистика – допълнителна функционалност**
  + Преглед на броя налични книги спрямо общия брой
  + Най-често наеманите книги
  + Най-активните наематели
  + Преглед на завършените наеми спрямо общия брой
* **Диаграма – допълнителна функционалност**
  + Диаграма, показваща месечния брой наеми

Приложението има и втора роля на достъп – потребители (в контекста на проекта наематели). Те имат достъп до следните следните функционалности:

* **Автентикация и сигурност**
  + Вход в системата (login)
  + Регистрация в системата (register)
  + Възстановяване на забравена парола (forgot password)
  + Нулиране на парола (reset password)
  + Промяна на паролата от профила на потребителя (change password)
* **Преглед на техните 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. Другият потребител, наемателят, има достъп до преглед на неговите наеми. 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 / потребителя чрез потребителския интерфейс. Те включват:

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

За да използва системата, администраторът / потребителят 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.

Потребителят също така може да се регистрира в системата. 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 или .png).

* **Данни за наематели**

Администраторът може да види следните параметри за всеки наемател:

* Потребителско 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 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 потребителските имена на наемателите. Има филтърната опция „търсачка“, като може да се търси по потребителско 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 – допълнителна функционалност**

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 – допълнителна функционалност**

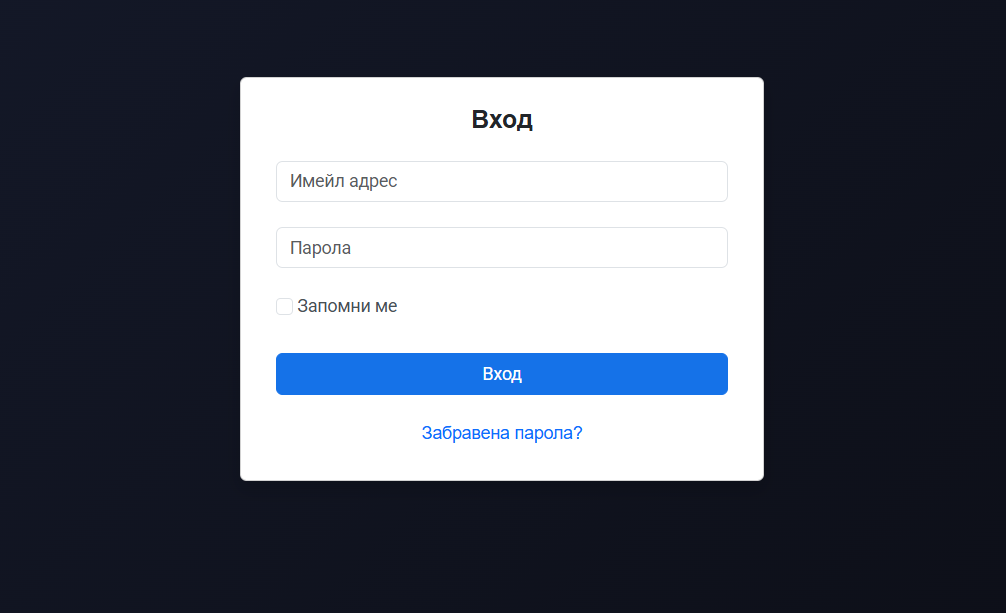
В началната страница администраторите могат видят диаграма, показваща месечния брой наеми.

* **Преглед на моите наеми**

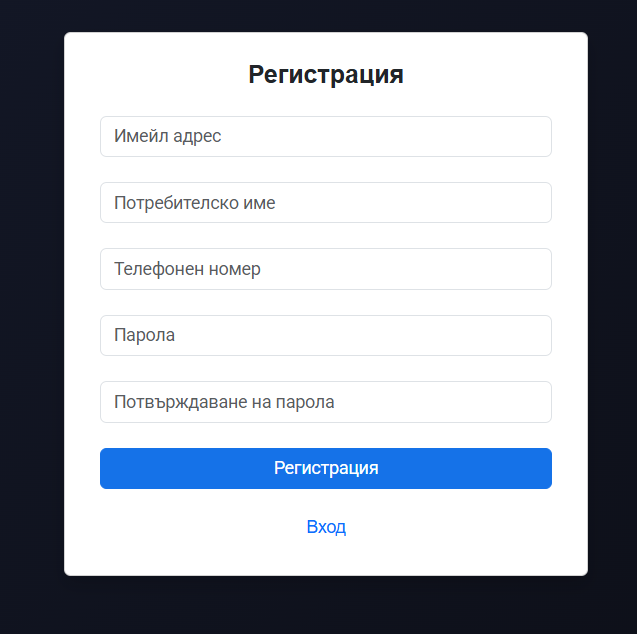
След като наемателят е влязъл в акаунта си, той е прехвърлен към страница, в която може да види наеми на негово име.

## Примерен потребителски интерфейс

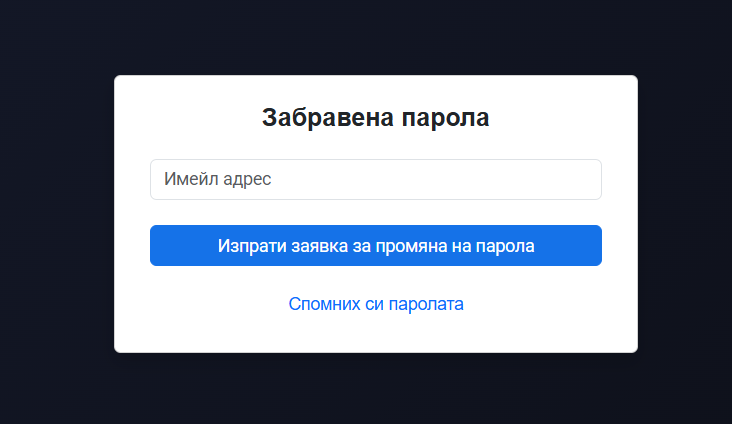
Графичният потребителски интерфейс (GUI) на уеб приложението е проектиран с цел улеснена навигация и интуитивно управление на книжния инвентар, наемателите и наемите. Визуалното оформление е съобразено с модерните уеб стандарти и използва чист и опростен дизайн, който позволява бърз достъп до основните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



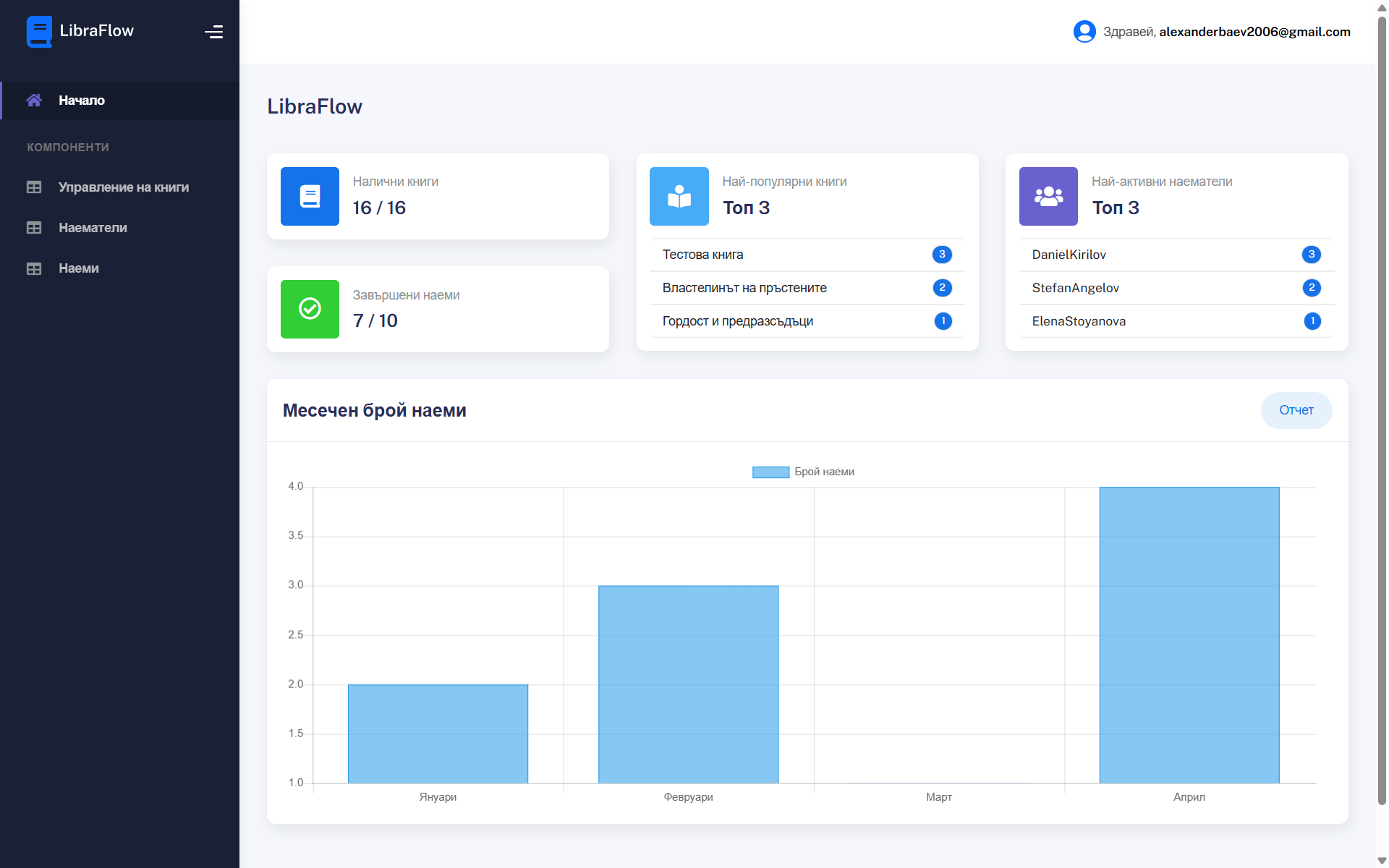
Фигура № 1: Вход



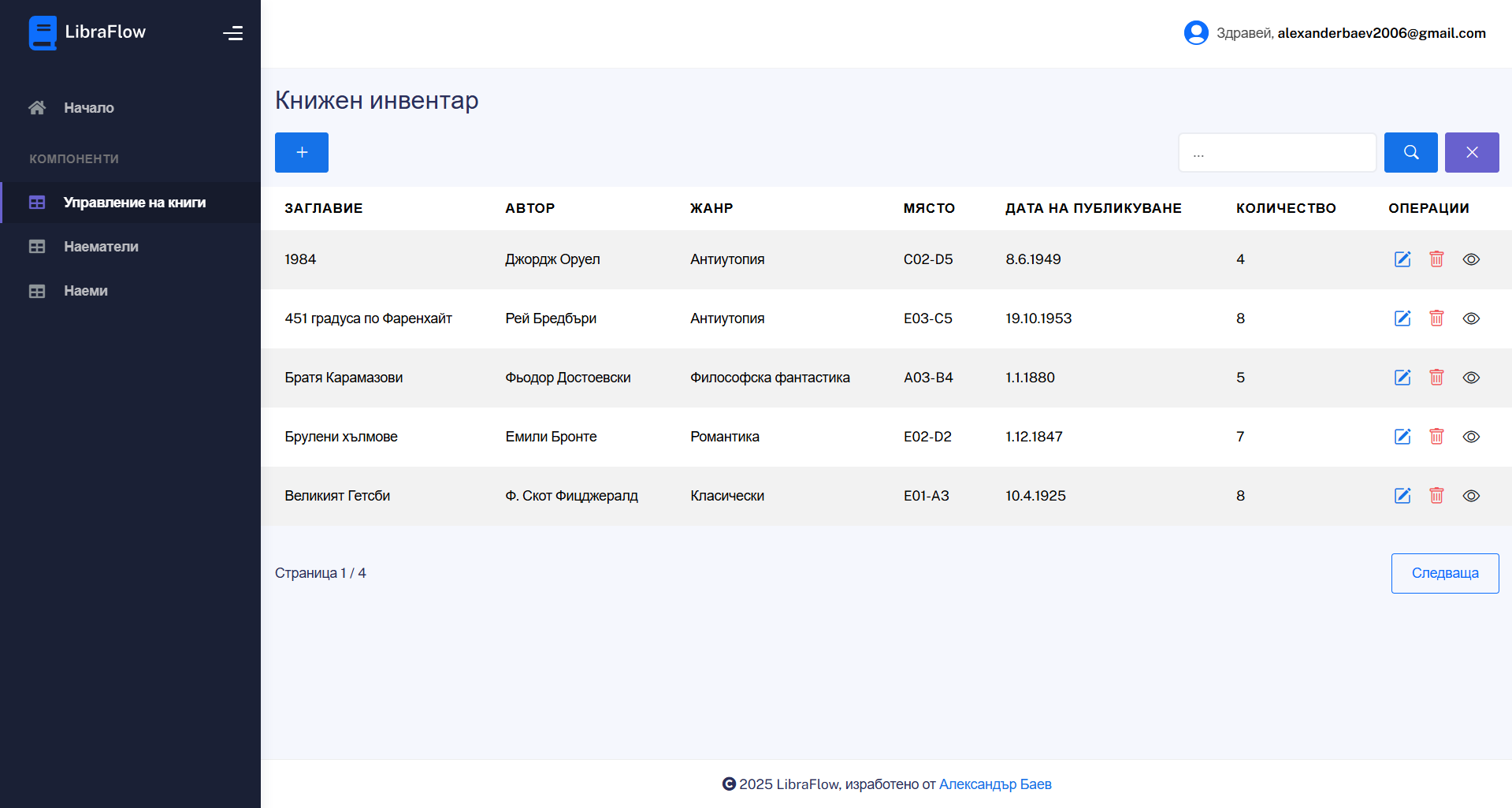
Фигура № 2: Регистрация



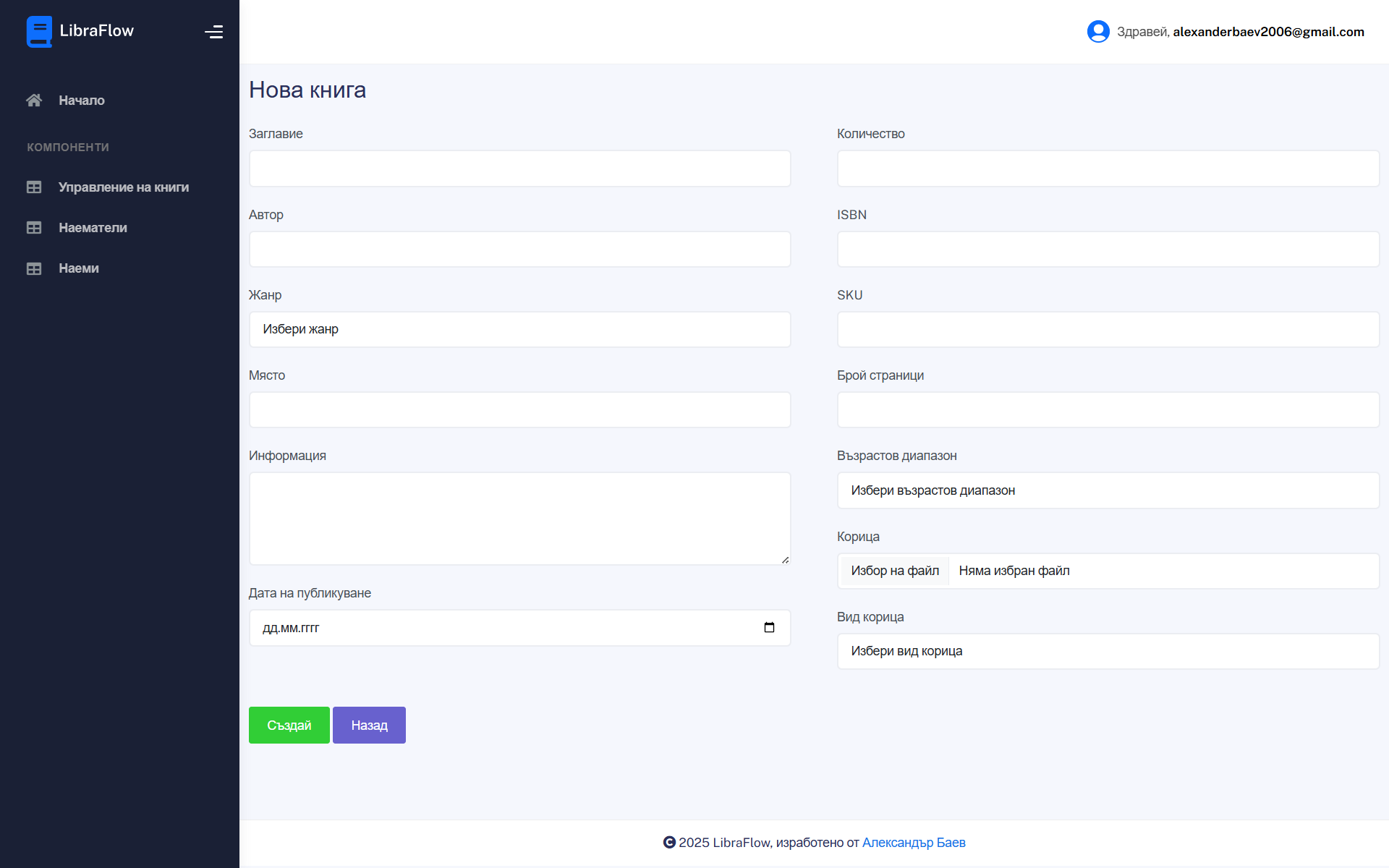
Фигура № 3: Забравена парола



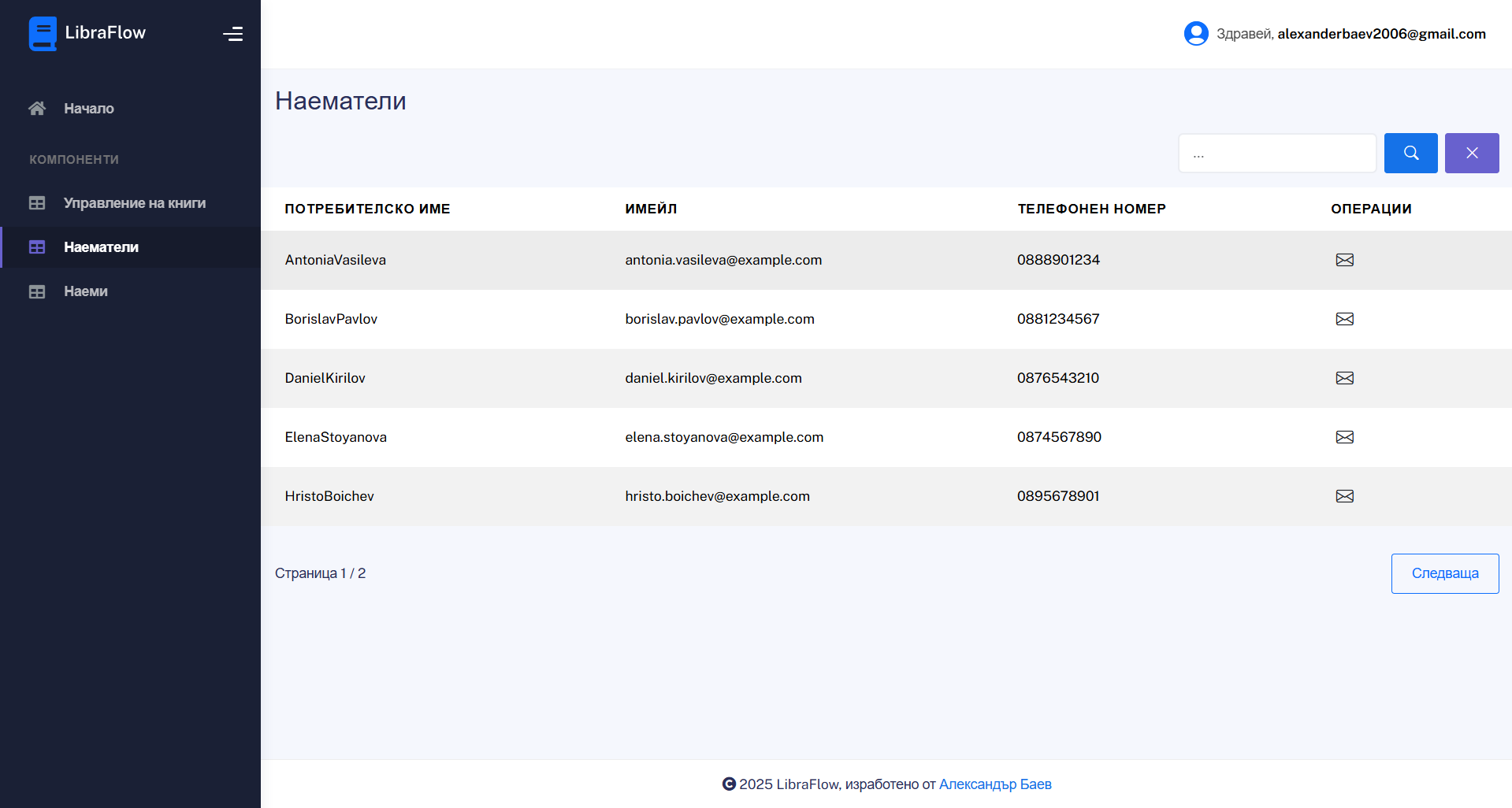
Фигура № 4: Начална страница



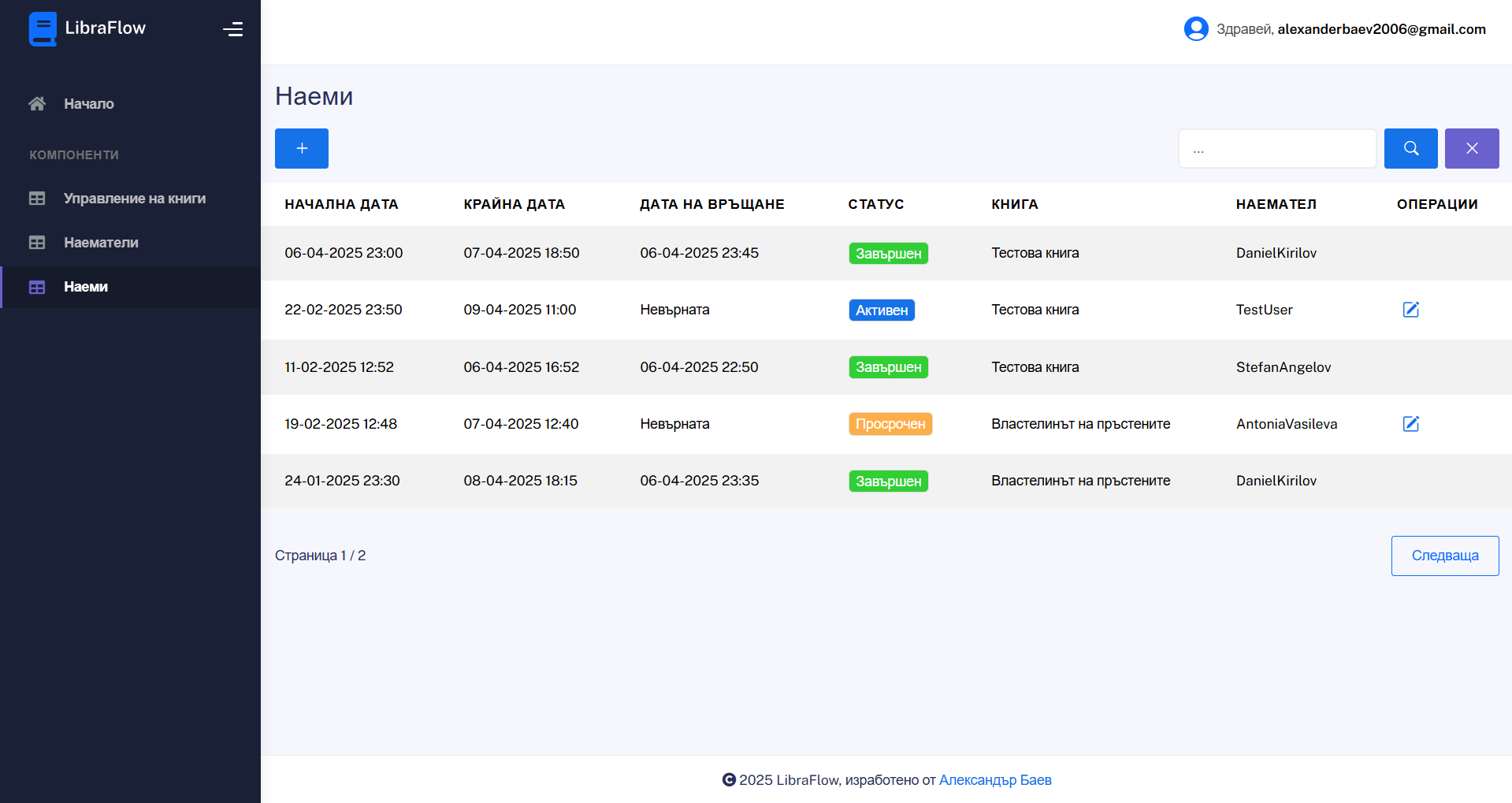
Фигура № 5: Книжен инвентар



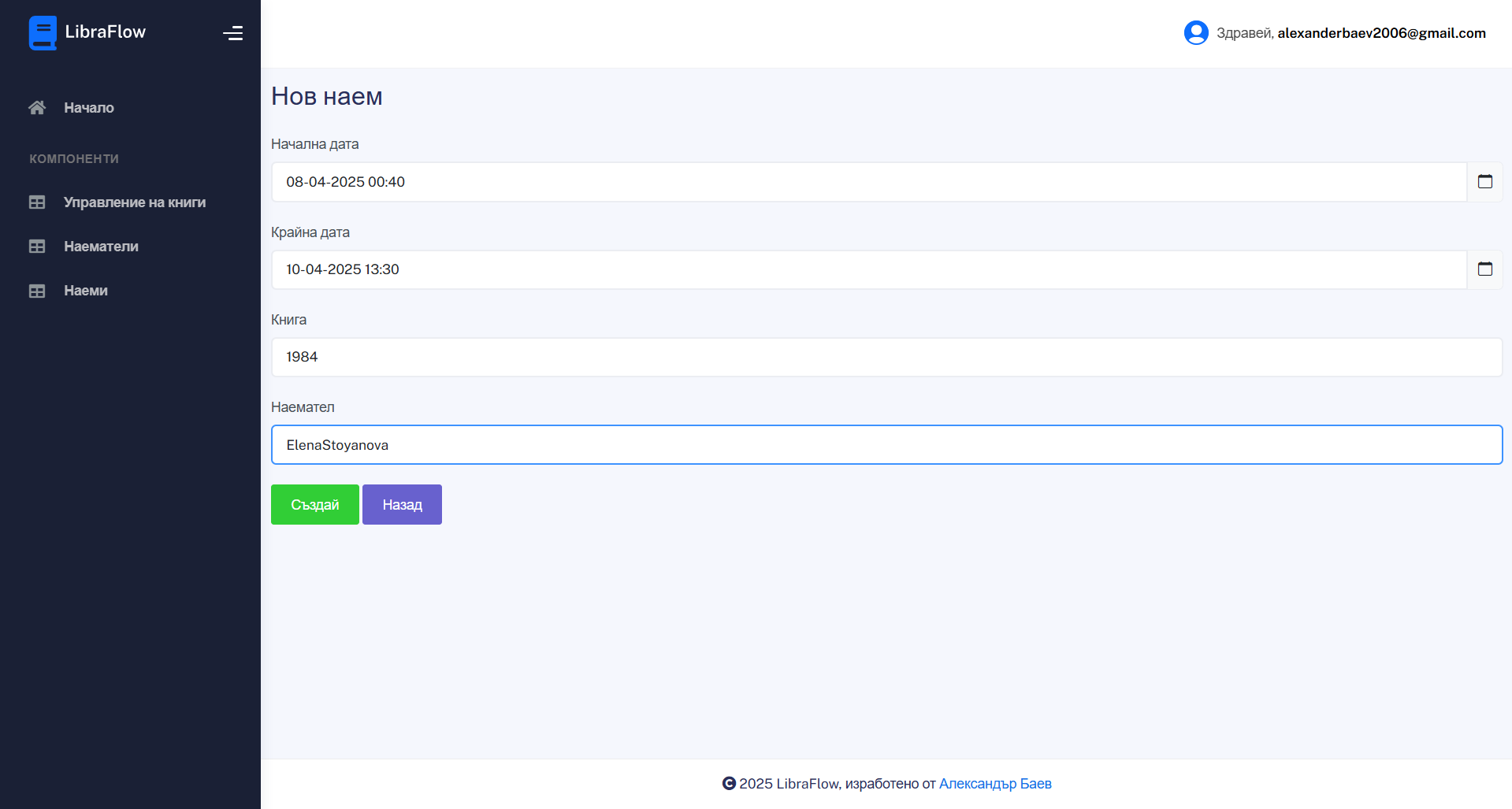
Фигура № 6: Добавяне на нова книга



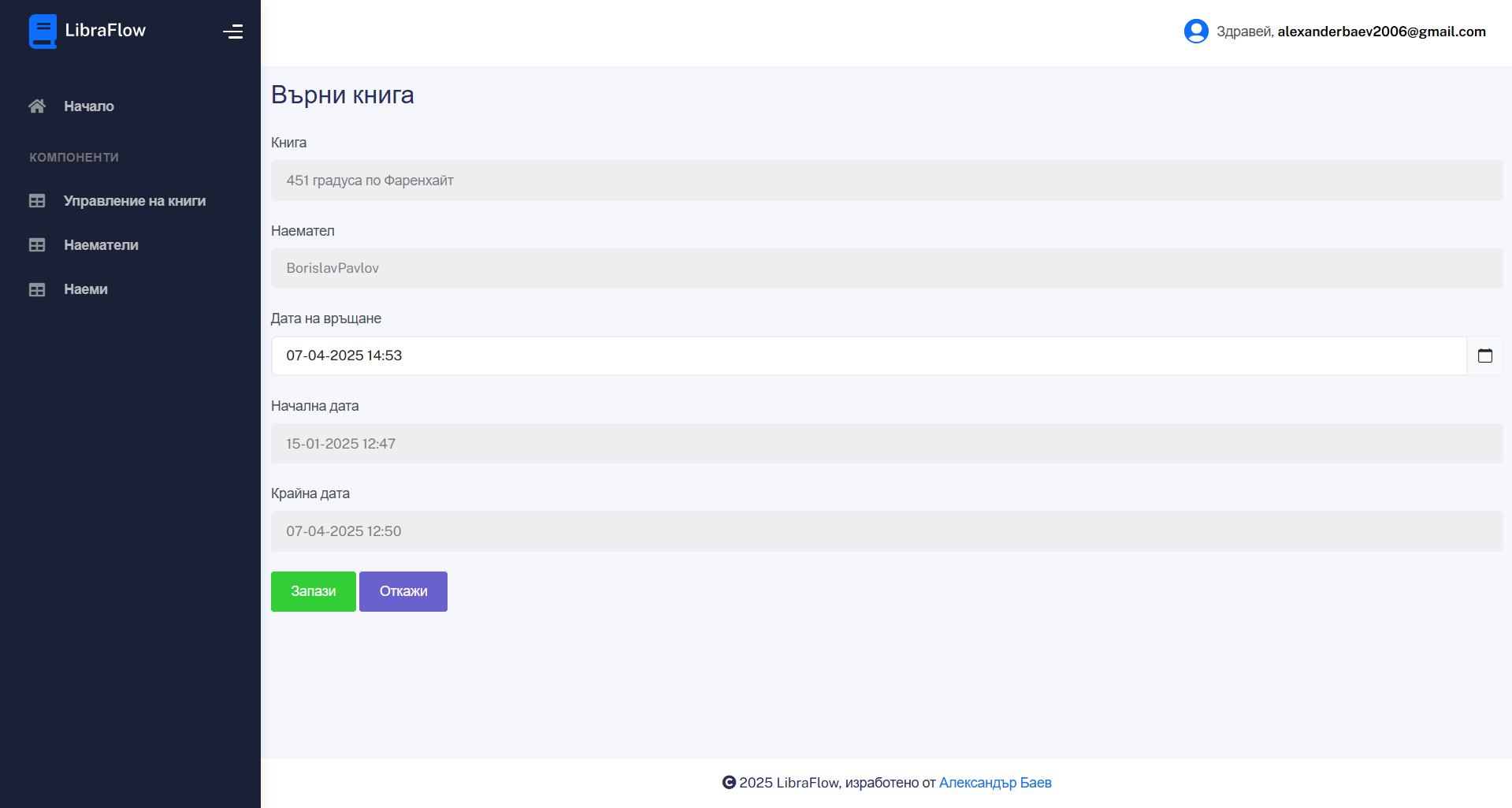
Фигура № 7: Наематели



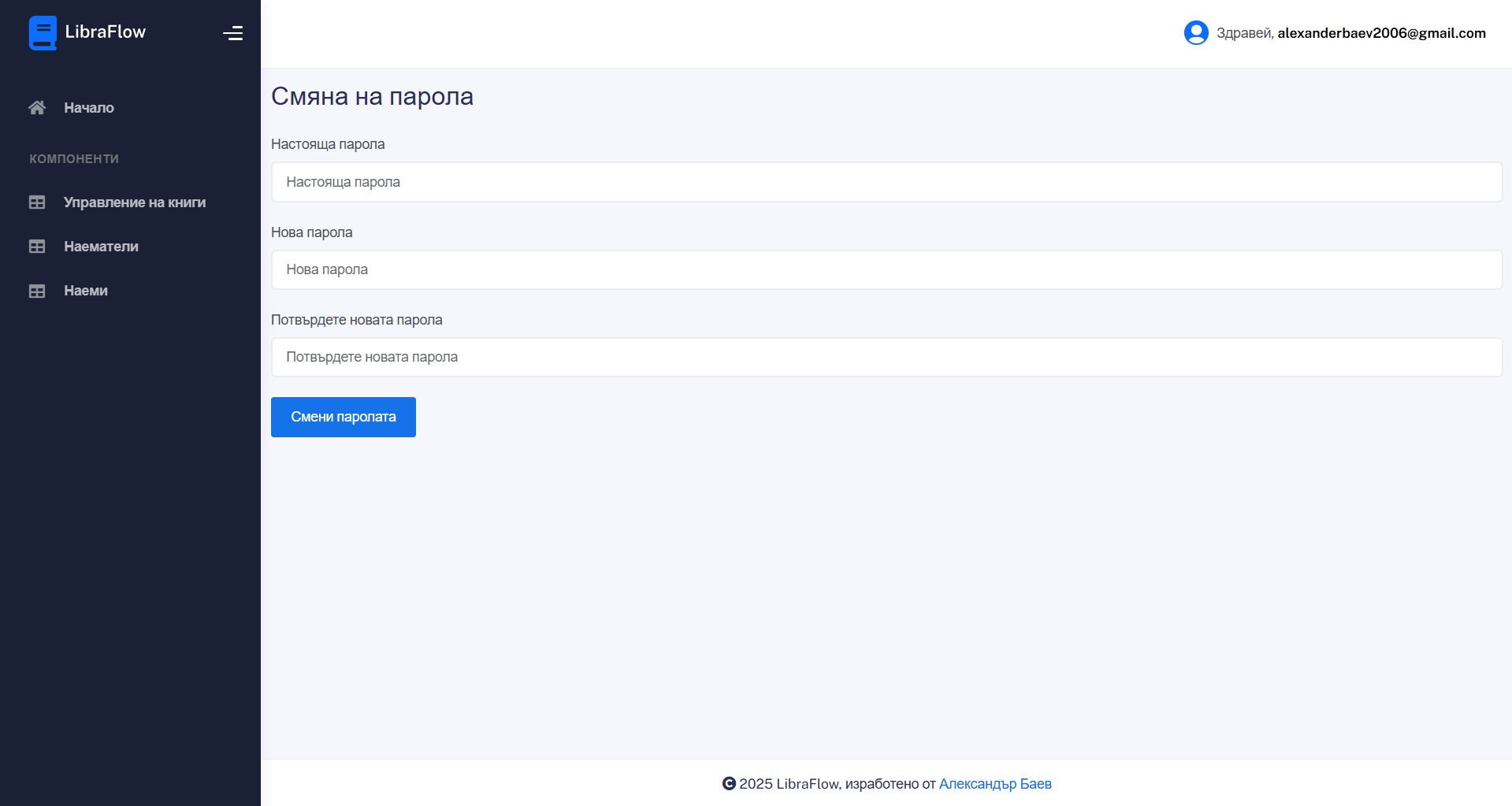
Фигура № 8: Управление на наеми



Фигура № 9: Създаване на нов наем



Фигура № 10: Връщане на книга



Фигура № 11: Смяна на парола



Фигура № 12: Генериране на PDF отчет

## 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. Чрез използването на Unified Modeling Language (UML**)** се осигурява стандартен подход за моделиране, който подпомага както процеса на разработка, така и бъдещото разширяване и поддръжка на системата.

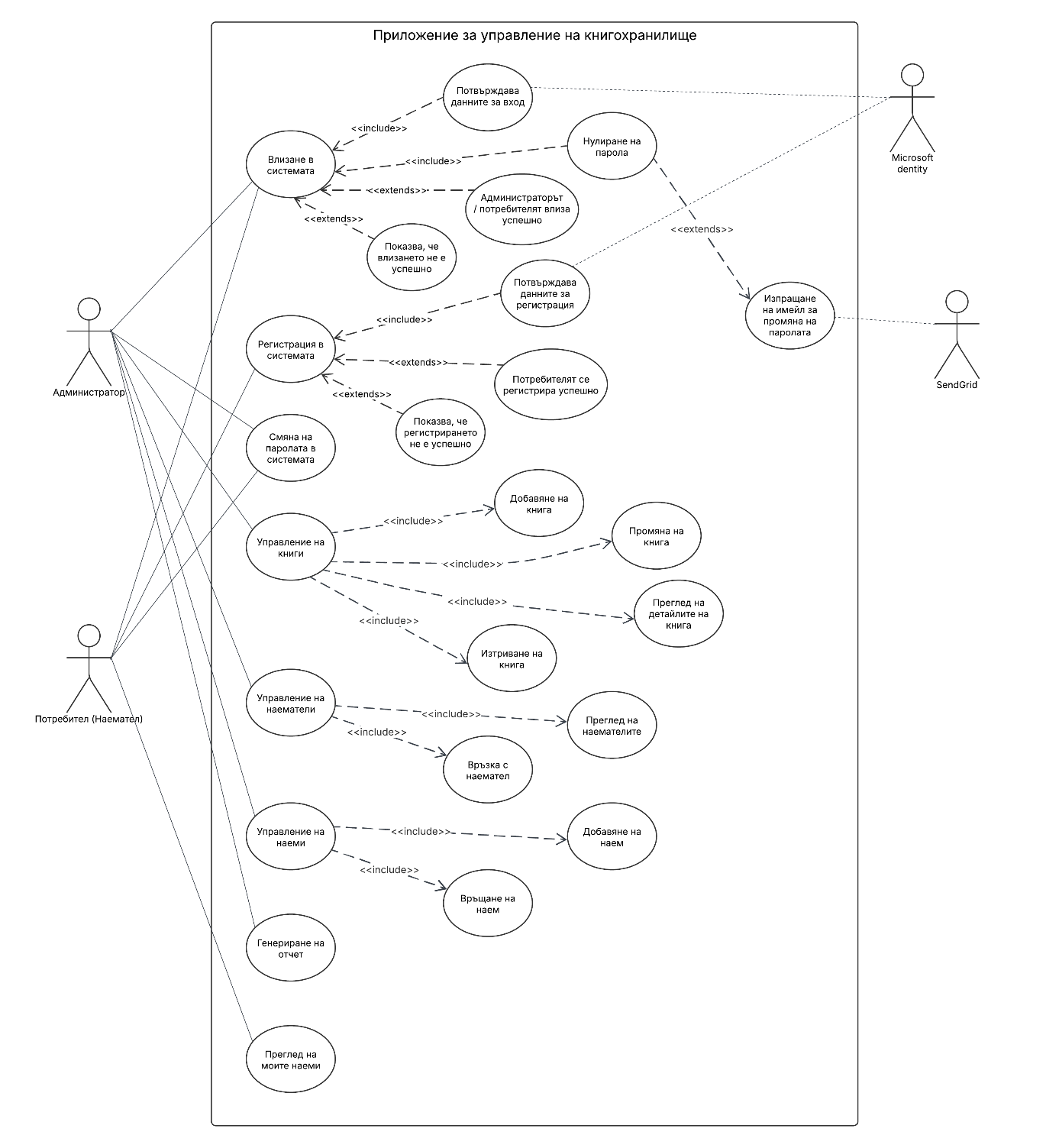
Всички диаграми са създадени в[Lucidchart](https://www.lucidchart.com/pages), като са добавенивGithub репозиторито на проекта.

**Диаграма на случаи за употреба** – Описва основните взаимодействия между администраторите, потребителите и системата.

На представената диаграма на случаите за употреба са изобразени основните взаимодействия между администраторът, наемателят и приложението. Администраторите разполагат с пълен контрол върху управлението на книги, наематели и наеми, докато наемателите имат достъп само до ограничена част от функционалността – по-конкретно до преглед на собствените си наеми.

Работният процес включва няколко ключови модула. И двете роли потребители могат да сменят тяхната парола. Системата предлага разширени възможности като интеграция със SMTP услуга (SendGrid) за изпращане на имейли и Microsoft Identity за сигурна автентикация.

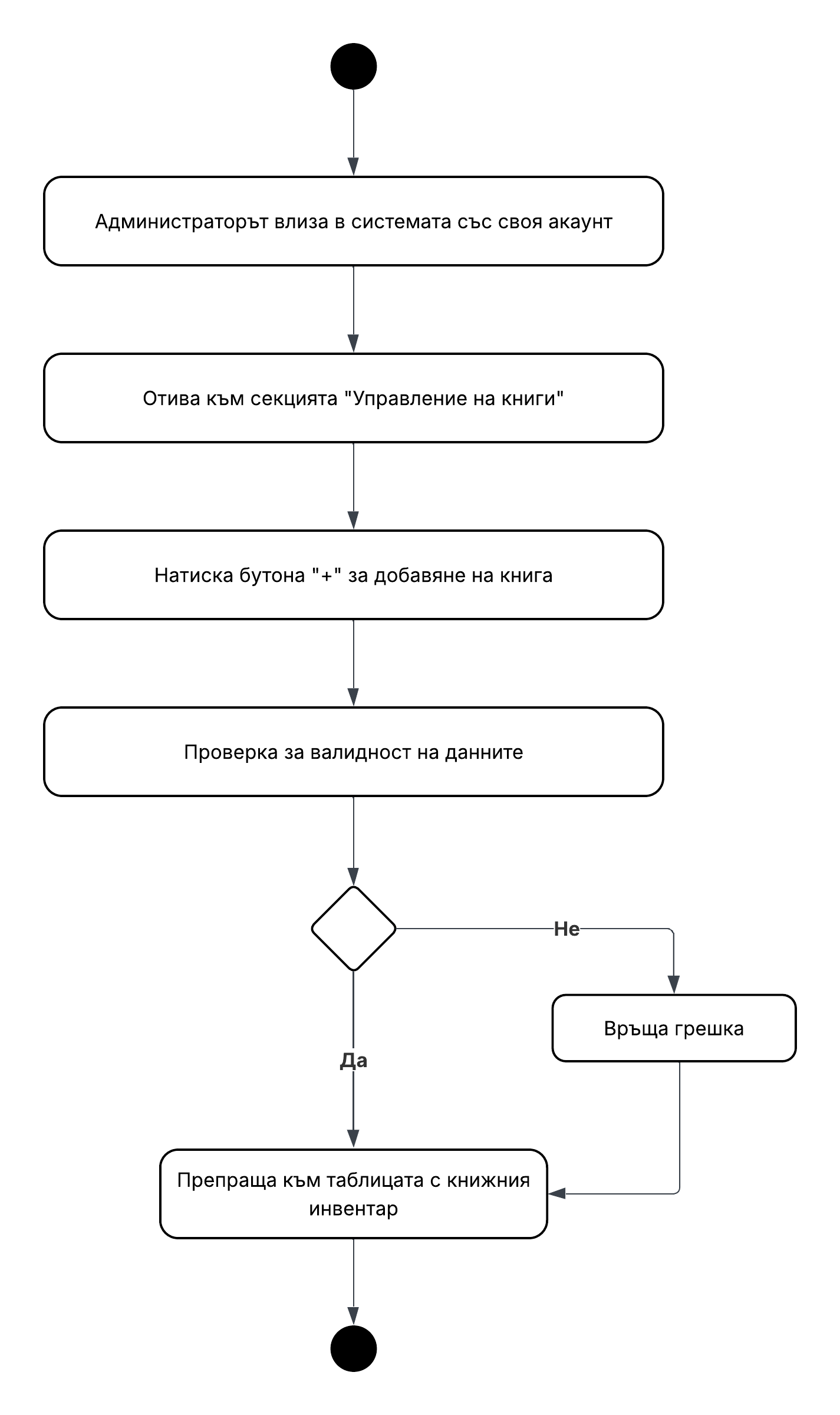
Диаграмата визуализира цялостния процес на управление на книгохранилище чрез ясно разделени функционални блокове. Всеки случай на употреба представя конкретно действие, което потребителят може да извърши, а използваните връзки <<includes>> и <<extends>> показват зависимостите между различните функции. Това позволява на системата да осигури гъвкава и ефективна платформа за управление на книжните ресурси, подобрявайки организацията.



Фигура № 13: Диаграма на случаите за употреба (Use Case)

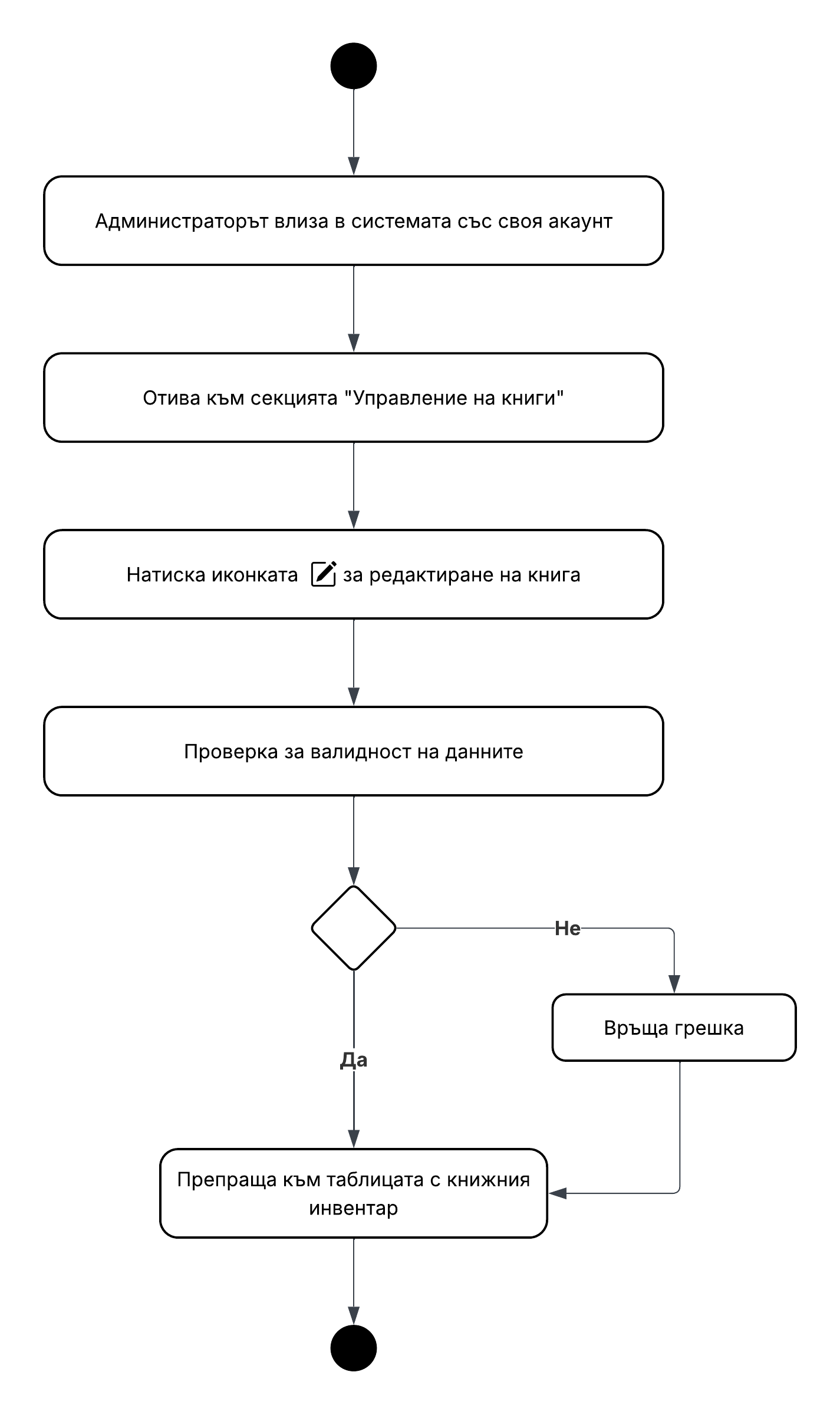
**Диаграми на дейностите** – представят логическия поток на различни процеси в системата

* **Добавяне на книга – стъпки на изпълнение:**



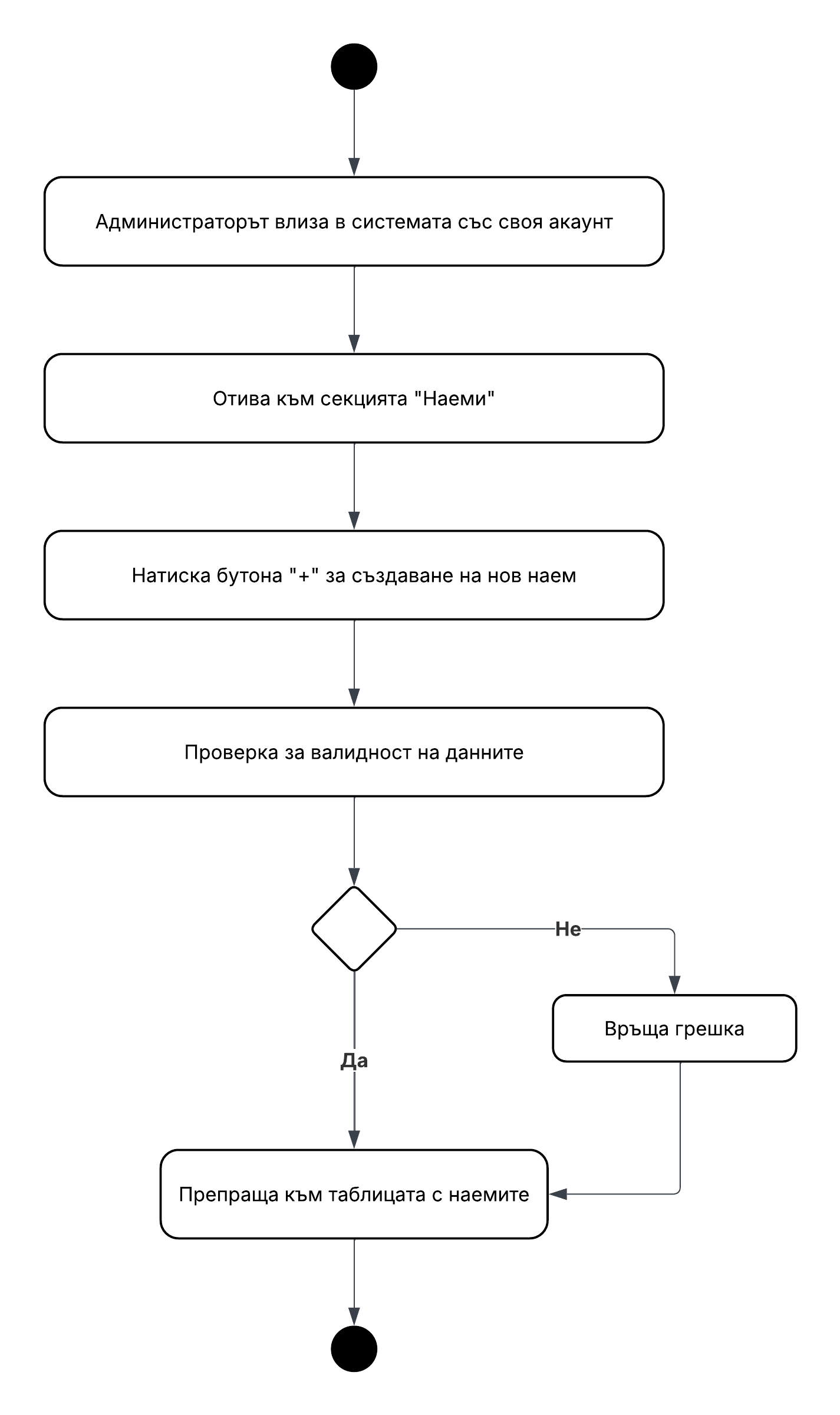
Фигура № 14: Диаграма на дейността №1 (Activity diagram)

* **Редактиране на книга – стъпки на изпълнение:**



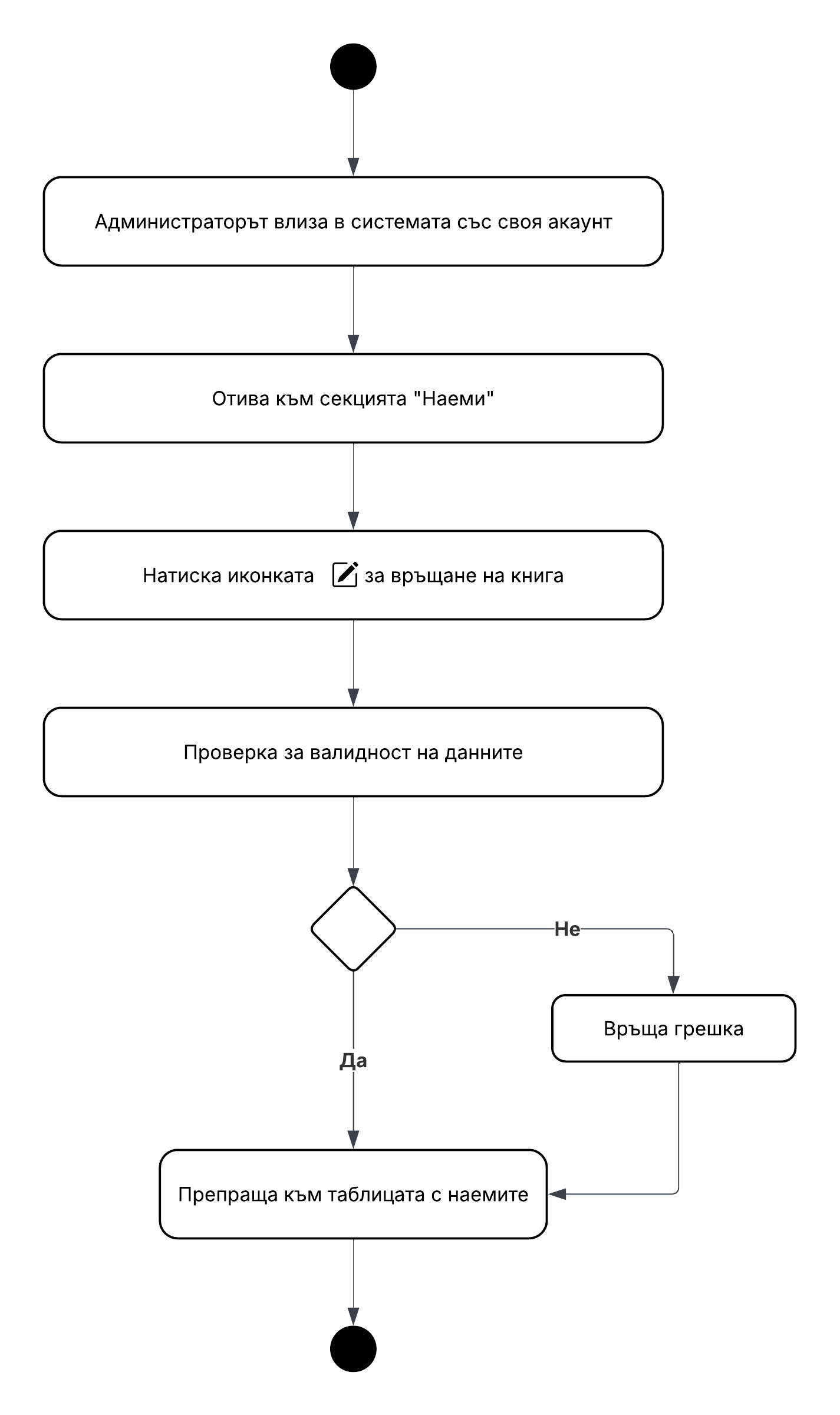
Фигура № 15: Диаграма на дейността №2 (Activity diagram)

* **Създаване на нов наем – стъпки на изпълнение:**



Фигура № 16: Диаграма на дейността №3 (Activity diagram)

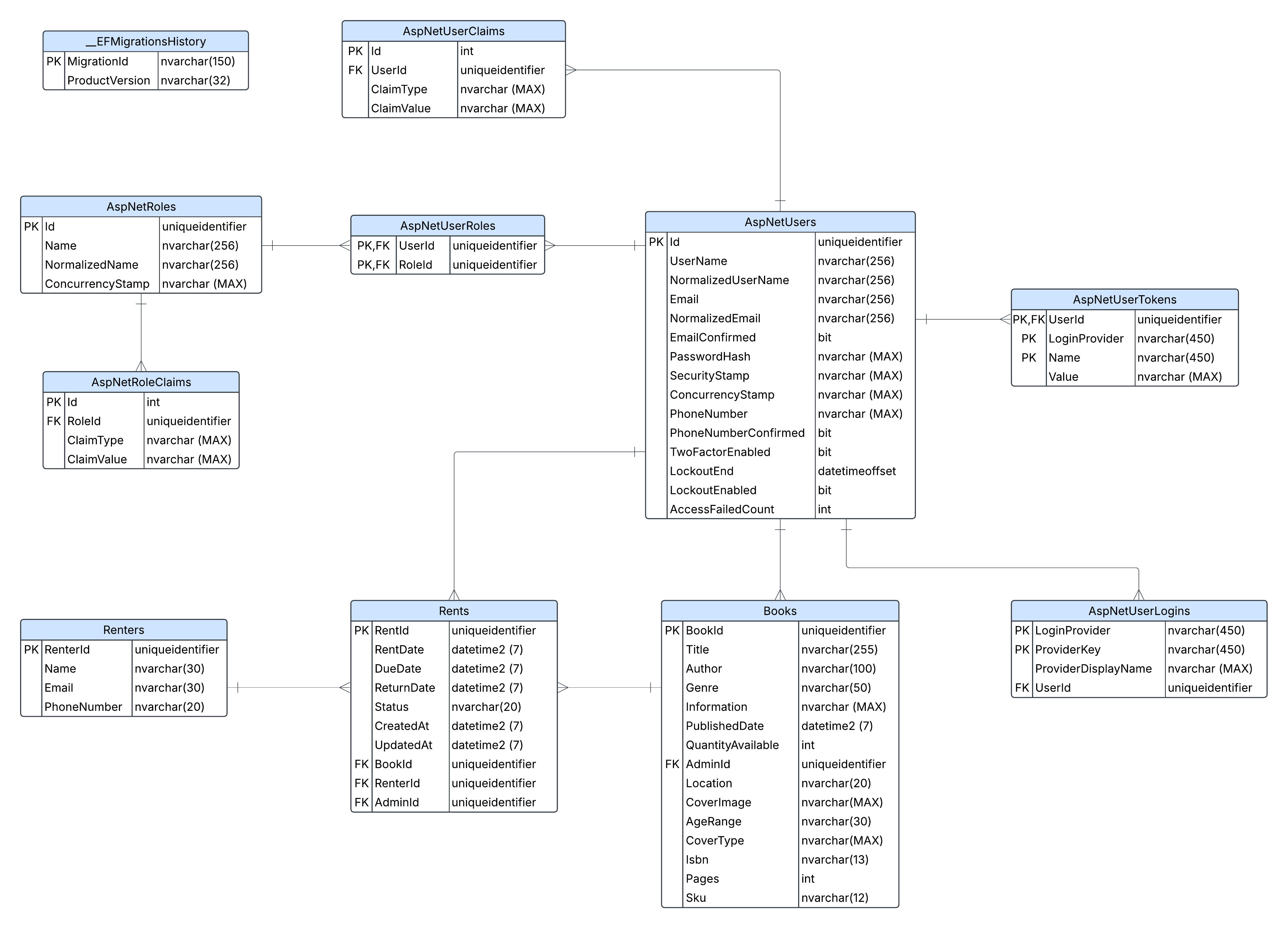
* **Връщане на книга – стъпки на изпълнение:**



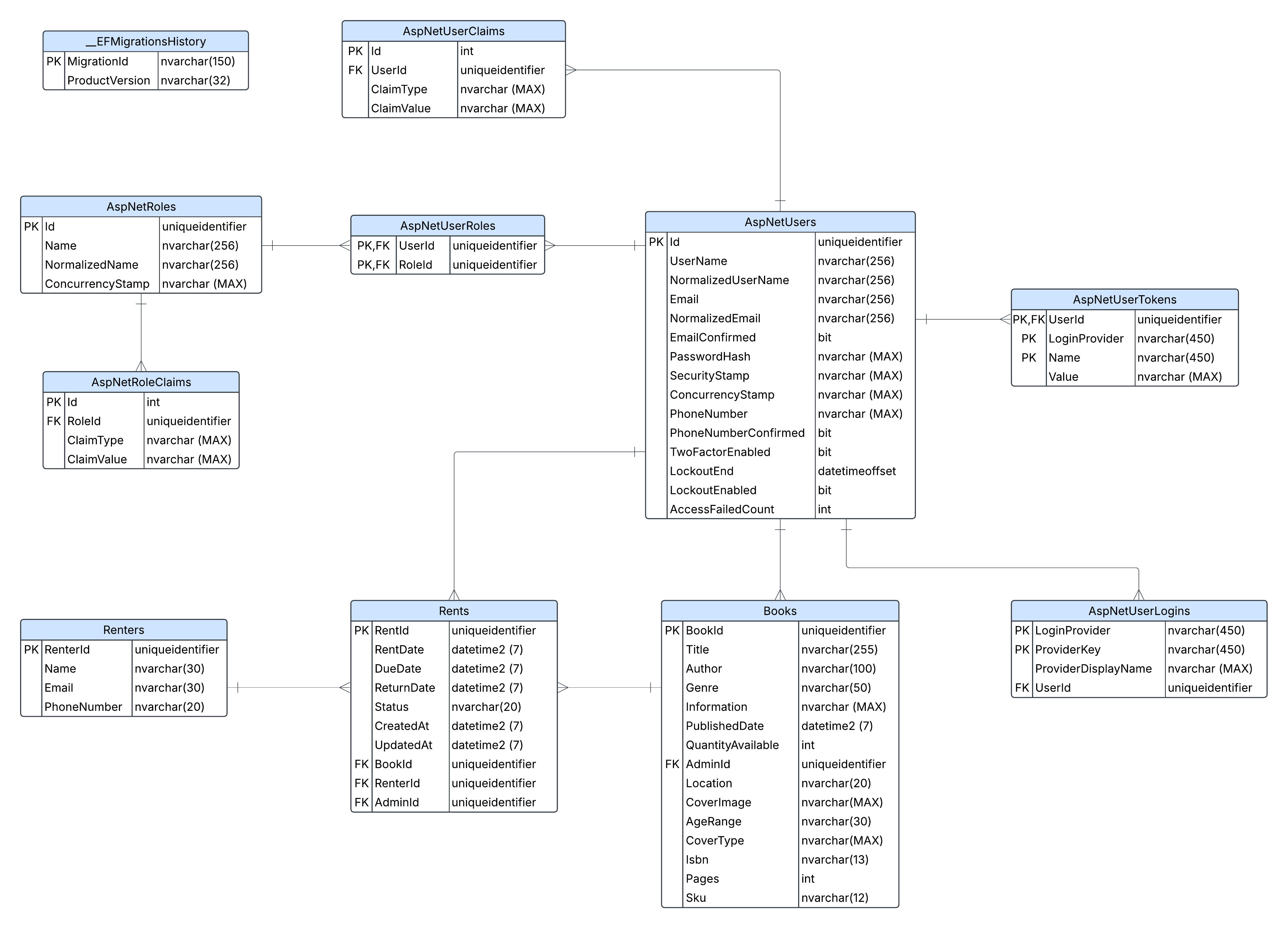
Фигура № 17: Диаграма на дейността №4 (Activity diagram)

## Модел на съдържанието / данните

Приложението за управление на книгохранилище разчита на добре дефиниран модел на данните, който комбинира текстови и графични ресурси. Основният фокус е върху управлението на книги, наематели и наеми, като всички данни се съхраняват в релационна база чрез Entity Framework Core. Графичните ресурси се съхраняват във формати, подходящи за уеб-базирани приложения (.png или .jpg).



Фигура № 18: Диаграма на отношенията между моделите (Entity Relationship)



* **Типове съдържание**

Текстово съдържание:

* **Потребителски данни (таблица AspNetUsers):** потребителско име, имейл, хеширана парола, телефонен номер, както и роля (Администратор или Наемател), зададена чрез AspNetRoles
* **Информация за книги (таблица Books):** заглавие, автор, описание, жанр, брой страници, дата на издаване, ISBN, SKU, място и тип корица.
* **Наеми (таблица Rents):** дата на започване, крайна дата, статус, връзка към книга и наемател.
* **Наематели (таблица Renters):** потребителско име, имейл, телефонен номер.

Графично съдържание:

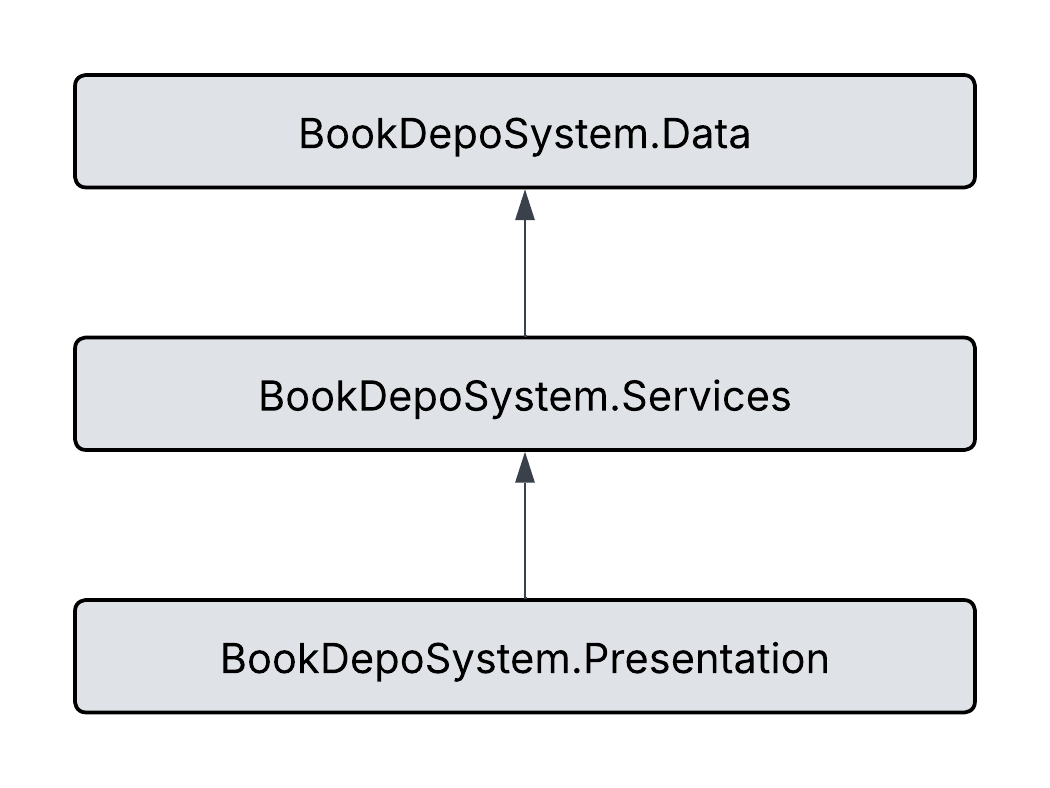
* **Корици на книги (Books → CoverImage):** изображенията се съхраняват като низове пътища към файлове (.png/.jpg) в сървърната директория. Поддържани формати: .jpg, .jpeg, .png. Препоръчителен максимален размер: 2MB.
* **Доклади (PDF):** всеки месечен отчет за наеми се генерира с QuestPDF и се предоставя за изтегляне. Файловете са във формат .pdf.
* **Структура и връзки**

ER-диаграмата показва ясно релационната структура между основните обекти в системата:

* **AspNetUsers –** основната таблица за автентикация и авторизация.
* **Renters** – наемателите са свързани с наемите чрез външен ключ (FK).
* **Books** – книгите имат връзка с администратора, който ги е добавил (AdminId).
* **Rents** – свързват книги и наематели, като всяко наемане съдържа информация за начална и крайна дата на връщане.
* **AspNetUserRoles**, **AspNetRoles**, **AspNetUserClaims** и свързаните таблици – управляват идентичността и ролите.
* **Характеристики и честота**
* Наемател може да има множество наеми
* Администратор може да добави множество книги и да създаде множество наеми.
* Книга може да участва в множество наеми (но само в един наем едновременно).

# Дизайн

Езикът за разработка е C# и проектът е основан върху .NET платформата. Използва се версия 8 за .NET както и за другите библиотеки, използвани в проекта. Разделен е на 3 слоя – слой за бизнес логиката, слой за достъпа до базите данни, презентационен слой и 2 допълнителни проекти – за тестове и общи файлове.



Фигура № 19: Диаграма на отношенията между главните проекти

* **Презентационен слой** – това е слоят, чрез който потребителя достъпва и работи с другите два слоя. Данните биват представяни в интерактивен и удобен за използване формат. Състои се от един главен проект:
* BookDepoSystem.Presentation – Това е основното Razor MVC приложение, което позволява на потребителите да извършват действия спрямо тяхната роля.
* **Слой за бизнес логиката** – това е слоят, в който се съдържа бизнес логиката в приложението. Свързан е със слоя за достъп до базите данни, за да може да работи с тях. В него има един проект:
* BookDepoSystem.Services – съдържа интерфейси и имплементации на основните услуги, свързани с управлението на книги, наематели и наеми. В този слой се обработват всички правила и операции, които не са пряко свързани с визуализацията или достъпа до базата данни, като например създаване на записи, валидиране на данни, проверка на срокове и автоматична актуализация на статуси.
* **Слой за данните** – в слоя за данните се съдържа инициализацията на базите данни, таблиците в нея и техни взаимовръзки. Има следния слой:
* BookDepoSystem.Data – слоят за данни на приложението, който съдържа контекстът и моделите. Работи с базата данни чрез Entity Framework Core (ORM библиотека).
* **В приложението се съдържат още четири проекта:**
* BookDepoSystem.Common – Съхраняват се в него локализирани думи, с цел превод на сайта на български език.
* BookDepoSystem.Tests – Този проект отговаря за тестването на проекта.

За да може приложението да бъде напълно функционално, се използват набор от зависимости(dependencies), които улесняват и подобряват разработката на софтуера:

• Entity Framework Core 8(EF Core 8) - ORM технология, която улеснява работата с базата данни. Проектът е изграден по принципа Code First, като първо се дефинират моделите, след което се генерират таблиците с необходимите полета, типове и релации

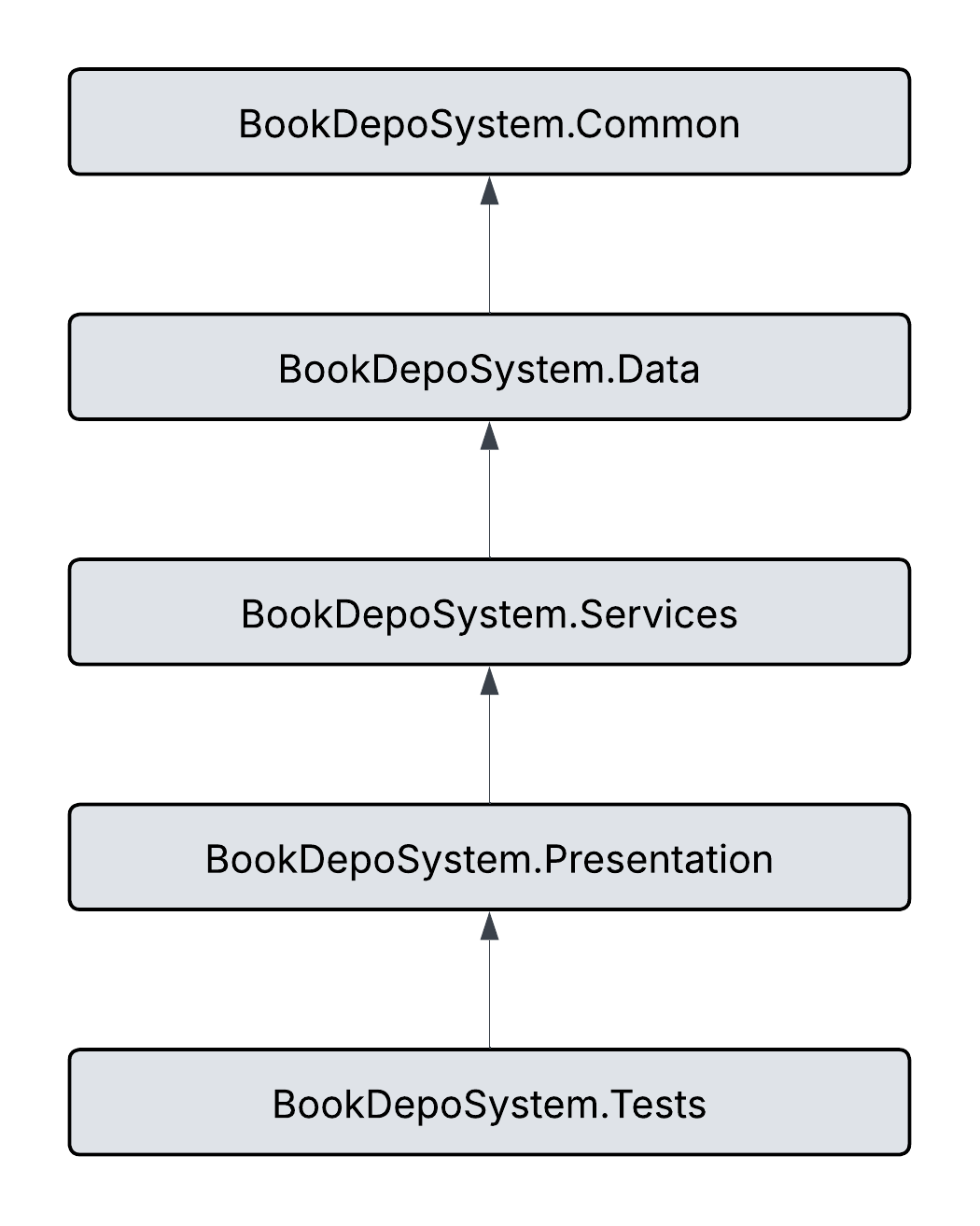
• EF Core 8 InMemoryDatabase – Разширение на EF Core 8, което позволява използването на временна база данни в оперативната памет, без необходимост от външен сървър. Подходящо е за тестване, тъй като методите на EF Core 8 трудно се заменят чрез стандартни mock библиотеки като Moq. Този метод се препоръчва от Microsoft като по-ефикасно решение вместо използването на фалшиви обекти.

• SendGrid – Инструмент за изпращане на имейли директно от приложението към потребителите.

• xUnit – Тестова рамка за модулно тестване на бизнес логиката и достъпа до базата данни.

• Stylecop Analyzers – Анализатор, който проверява C# кода за съответствие с установени стилови и структурни правила

## Реализация на архитектурата на приложението

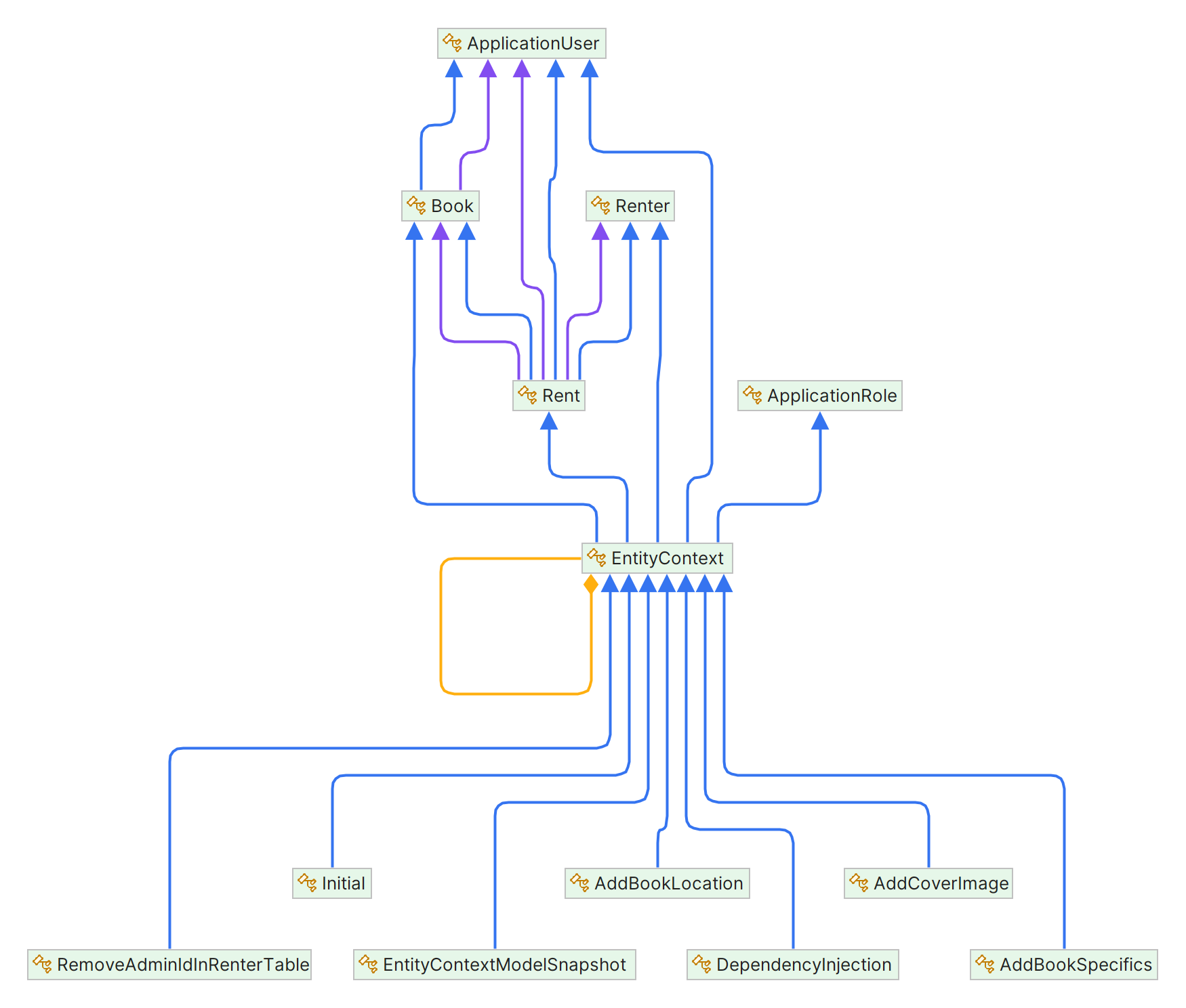


Фигура № 20: Диаграма на отношенията между проектите и техните зависимости

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

Приложението е изградено от следните слоя:

* **Слой за достъп до данните:**



Фигура № 21: Диаграма на слоя от данни

Диаграмата представя слоя за достъп до данни в приложението, който е реализиран чрез Entity Framework Core. Основната му цел е да осигури взаимодействие между приложението и базата данни чрез дефинирани модели и контекст.

* + **Описание и предназначение:**

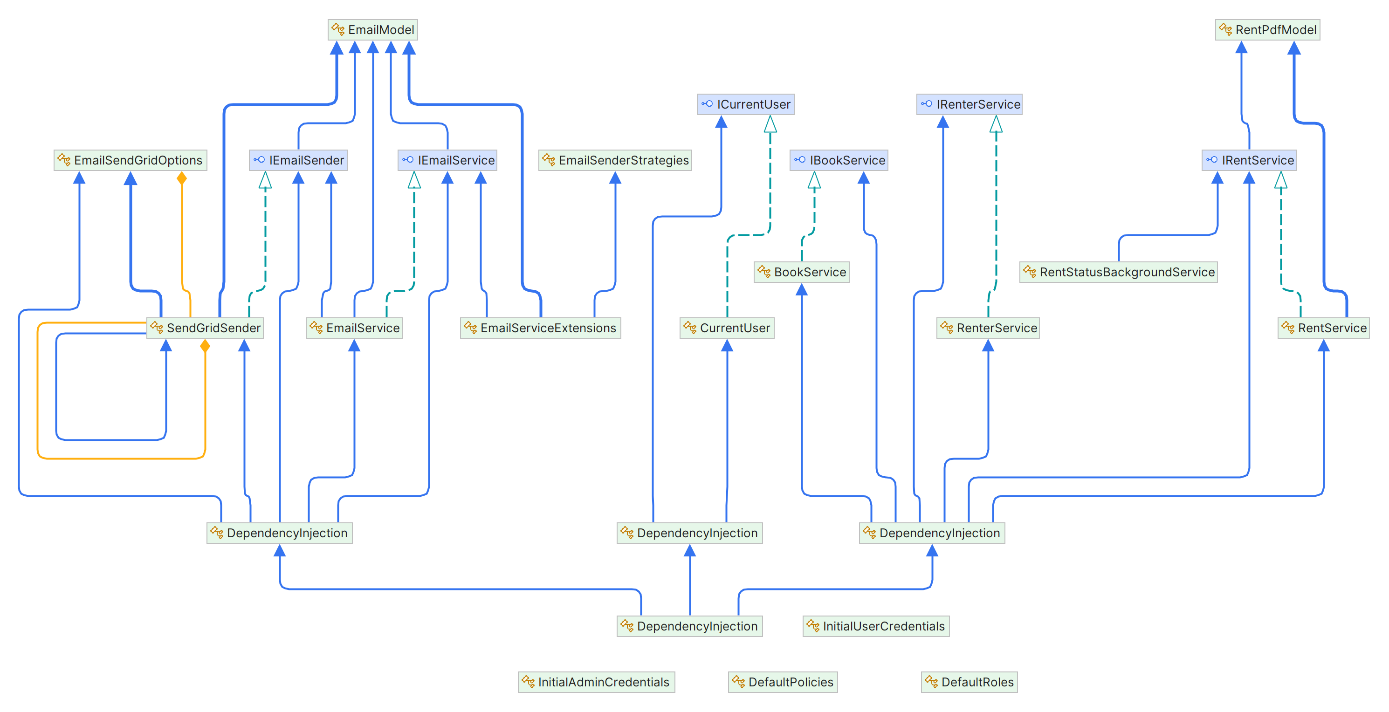
Сърцето на този слой е EntityContext – това е класът, който наследява DbContext и отговаря за управлението на връзката с базата данни. В него се намират всички DbSet<> свойства, представляващи таблиците: ApplicationUser, ApplicationRole, Book, Renter, Rent и др.

Свързаните модели като Book, Renter, Rent и ApplicationUser съдържат нужната структура и релации, които се използват за автоматично създаване и поддръжка на схемата на базата данни чрез миграции.

* + **Основни компоненти:**
    - ApplicationUser и ApplicationRole – разширения на ASP.NET Identity моделите, използвани за удостоверяване и управление на роли.
    - Book и Renter – модели, описващи книги и наематели, които имат релации с наемите (Rent).
    - Rent – таблица, която свързва книги и наематели с конкретни периоди на наем.
    - Миграции (Initial, AddBookLocation, AddCoverImage, и др.) – използвани за контрол на версиите на базата данни, позволяват проследяване и прилагане на промени в структурата.
  + **Предназначение:**

Този слой капсулира логиката за съхранение, извличане и манипулация на данни, осигурявайки абстракция над базата данни. Той позволява на бизнес логиката да използва обектно-ориентирани модели без директно писане на SQL заявки, като същевременно поддържа свързаност и целостта на данните.

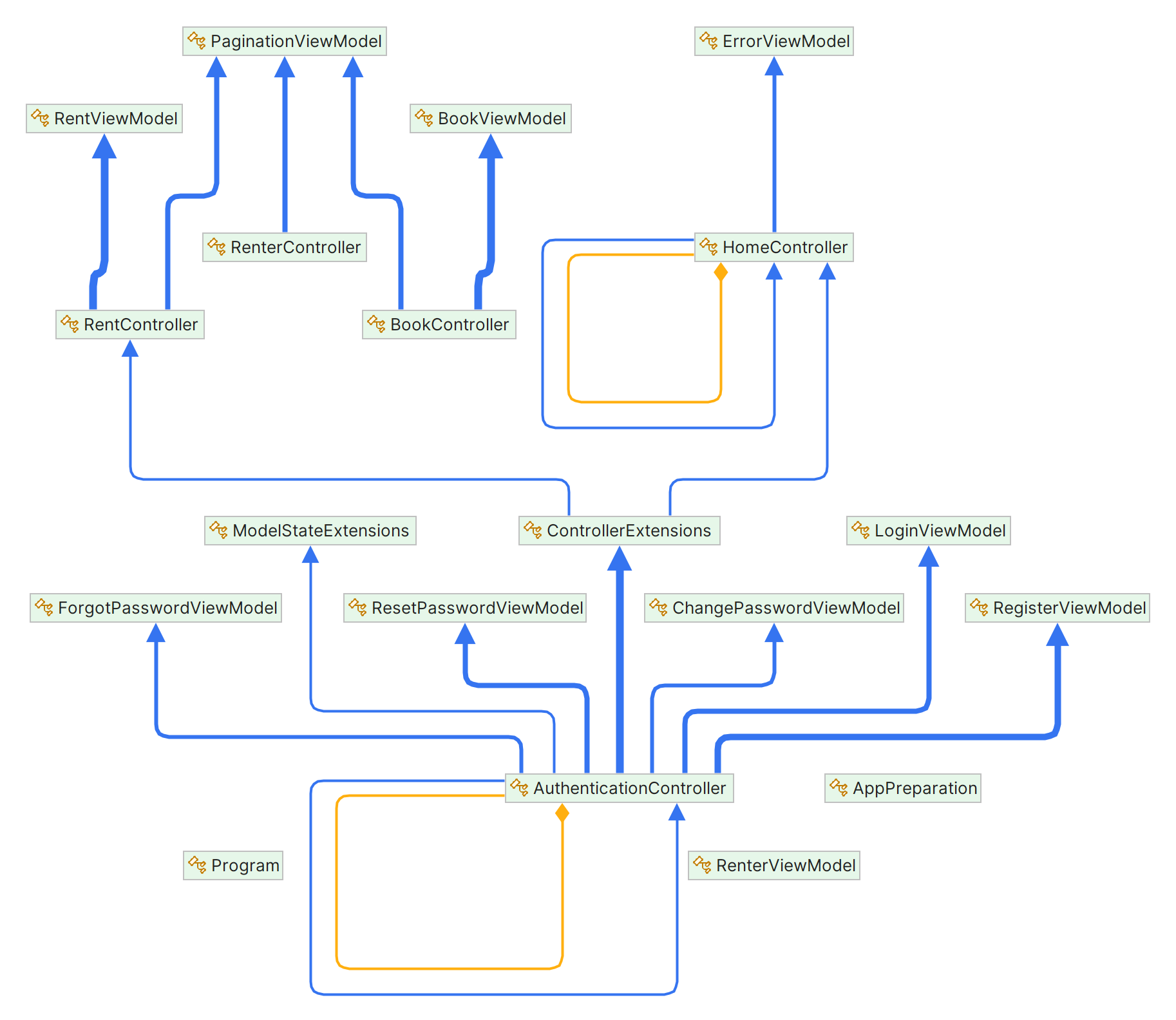
* **Слой на бизнес логиката:**



Фигура № 22: Диаграма на бизнес слоя

Този слой е представен от проекта BookDepoSystem.Services. Той съдържа основните бизнес услуги и дейности на потребителя. Разделен е на няколко именни пространства и папки.

* + **Common** - Съдържа споделени ресурси и помощни класове, използвани в други части на проекта
  + **Contracts** - Съдържа публичните интерфейси на услугите, които се използват в презентационния слой (UI). Тези интерфейси дефинират операциите, свързани с управление на книги, наематели и наеми.:
    - IBookService
    - IRenterService
    - IRentService
  + **Identity** - Съдържа функционалности, свързани с потребителите и сигурността:
  + **Implementations** - Съдържа реалните имплементации на услугите, описани в Contracts:
    - BookService
    - RenterService
    - RentService
    - RentStatusBackgroundService
  + **Models -** Съдържа модели, използвани вътрешно в бизнес логиката
  + **DependencyInjection -** Клас, който реализира инжектиране на зависимости от този слой към останалите.
* **Презентационен слой:**

****

Фигура № 23: Диаграма на презентационния слой

Този слой е представен от проекта BookDepoSystem.Presentation. Той отговаря за визуалната репрезентация на приложението. Разделен е на няколко имеви пространства и папки.

* + Controllers – В това именно пространство се съдържат класовете за управление(controllers), които работят с услуги от бизнес слоя.
  + Extensions - В това именно пространство се съдържат разширения, използвани от презентационния слой.
  + Models – В това именно пространство се съдържат моделите, които заместват моделите от базата в класовете за управление.
  + Views – В това именно пространство се съдържат изгледите/потребителския интерфейс на приложението, които работят с класовете за управление.

## Организация и код на заявките към база от данни

Достъпът до базата от данни се реализира чрез Entity Framework Core – ORM (Object-Relational Mapping) библиотека, която позволява работа с базата чрез C# обекти, без нужда от ръчни SQL заявки.

Цялата комуникация с базата се осъществява чрез класа EntityContext, който наследява DbContext. В него са описани всички таблици под формата на DbSet<T>:

* DbSet<Book> Books
* DbSet<Renter> Renters
* DbSet<Rent> Rents
* DbSet<ApplicationUser> Users

Методите за работа с базата от данни са разположени в **сервизния слой** (Services) и използват асинхронни заявки чрез LINQ. Основните операции включват:

* **Извличане на данни:**  
  Използват се методи като ToListAsync(), FirstOrDefaultAsync(), Where() за филтриране и извличане на обекти.
* **Добавяне на обекти:**  
  Чрез метода AddAsync() и SaveChangesAsync():
* **Изтриване на обекти:**  
  Чрез метода Remove():
* **Обновяване на обекти:**  
  Обектите се маркират като модифицирани и след това се извиква SaveChangesAsync():

В проекта се прилага **Repository-like pattern** чрез услуги, които инкапсулират достъпа до базата и разделят логиката на слоеве.

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

Приложението разполага с уеб базиран потребителски интерфейс, реализиран с помощта на **ASP.NET Core MVC**. Интерфейсът е достъпен чрез уеб браузър.

**Основни функционалности на потребителския интерфейс:**

* **Автентикация и сигурност:**
  + Вход в системата (login)
  + Регистрация в системата (само за потребителите)
  + Възстановяване на забравена парола (forgot password)
  + Нулиране на парола (reset password)
  + Промяна на паролата от профила на администратора (change password)
* **Начална страница –** В нея администраторът може да генерира PDF подробен месечен отчет за наемите
* **Управление на книги –** Администраторът може да добавя, редактира и да изтрива книги. Също така може да види детайлите на всяка книга.
* **Управление на наематели –** Администраторът може да вижда всички регистрирани наематели и да им пише по имейл.
* **Управление на наеми –** Администраторът може да добавя наеми, да следи статуса на наемите и да задава дата на връщане на наемите.

Допълнителни функционалности:

* **Статистика**
  + Преглед на броя налични книги спрямо общия брой
  + Най-често наеманите книги
  + Най-активните наематели
  + Преглед на завършените наеми спрямо общия брой
* **Диаграма –** показва месечния брой наеми
* **Търсачка** при страниците за управление на книги, наематели и наеми

Интерфейсът използва Razor страници и Bootstrap за визуално оформление.

# Ефективност и бързодействие на решението

Разработеното уеб приложение за управление на книжен депозит е проектирано с фокус върху добра производителност и ефективно използване на ресурси. Решението използва утвърдени технологии и подходи за изграждане на съвременни уеб приложения, което допринася за неговата стабилност и бързодействие.

* **Анализ на производителността:**
  + При тестове в среда с реалистични данни (над 500 книги и 100 наематели) системата отговаря с време под 500ms за основните операции като зареждане на таблици и извършване на CRUD действия.
  + Генерирането на PDF отчети с помощта на **QuestPDF** се изпълнява локално, без външен API, като времето за рендиране на един отчет е под 1 секунда ([QuestPDF, 2024](https://www.questpdf.com/" \t "_new)).
* **Сравнение с алтернативни решения:**

Съществуват подобни платформи и решения за управление на библиотечни системи, като Koha и Ex Libris. Те обаче са значително по-големи и по-тежки, с по-сложна конфигурация и често ненужни функционалности за малък книжен депозит. Разработеното решение е леко, фокусирано и съобразено с конкретните нужди на организацията, което води до по-добро време за реакция и по-малък learning curve за администраторите.

# Тестване

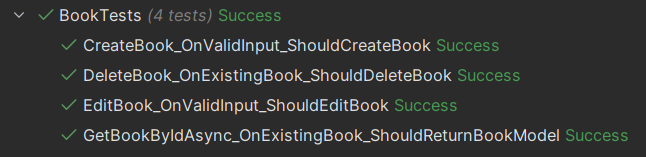
В този раздел са описани тестовите случаи и видовете тестване, които са предвидени за реалното изпълнение на проекта. Тестването включва:

**Видове тестове**

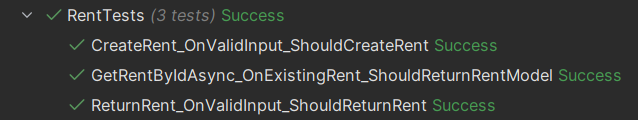
* **Unit тестове** – покриват основните функционалности на системата, като управление на книгохранилищните инвентари и наемите.
* **Интеграционни тестове** – проверяват взаимодействието между различните модули, включително генериране на отчети.

**Unit тестове**

* **Тестове за управление на книгохранилищни инвентари**



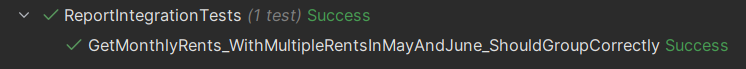
* Тест за извличане на книга по ID – проверява дали системата успешно връща информация за съществуваща книга.
* Тест за създаване на книга – проверява дали книга може да бъде създадена успешно и данните ѝ да се запазят в базата данни.
* Тест за редактиране на книга – тества дали промените в съществуваща 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 рамките на настоящия проект беше разработено уеб приложение за управление на книжен депозит, предназначено за администратори и наематели. Решението обхваща пълния жизнен цикъл на работа с книги – от създаване и редактиране, през управление на наематели, до проследяване на наеми и генериране на справки. Внедрени са функционалности като потребителско управление, ролеви достъп, отчетност и визуализация на данни.

* **Предимства на използваните технологии:**
  + **ASP.NET Core** предлага висока производителност, добра поддръжка и стабилност.
  + **Entity Framework Core** улеснява достъпа до база данни чрез ORM подход и дава възможност за лесно изграждане на заявките с LINQ.
  + **QuestPDF** е ефективна и лека библиотека за генериране на PDF документи без нужда от външни услуги.
  + **Слоестата архитектура** улеснява поддръжката, модулността и разширяемостта на приложението.
* **Ограничения:**
  + Решението е разработено основно за уеб среда и не поддържа офлайн режим.
* **Възможни алтернативи и тяхното сравнение:**
  + **Blazor** може да бъде използван за подобряване на интерактивността чрез уеб компоненти, но увеличава сложността.
  + **React или Angular** биха предоставили по-модерен фронтенд, но изискват повече ресурси и разделяне на проекта.
* **Практическа приложимост:**

Подобни решения намират широко приложение в библиотеки, училища, читалища и частни архиви, където е необходимо управление на физически книги и проследяване на техния статус. Проектът е лек, разбираем и може лесно да се адаптира към различни институции.

* + Възможности за бъдещо развитие:
  + Добавяне на потребителски портал за наематели, чрез който да преглеждат историята на наемите си и да подават заявки.
  + Поддръжка на известия по имейл при просрочени наеми.
  + Мобилна версия или Progressive Web App (PWA).
  + Интеграция с външни каталози или системи (например ISBN API) за автоматично попълване на информация за книги.
  + Аналитичен модул за проследяване на най-често наемани книги и активност на наемателите.

В заключение, проектът е стабилна основа за реална система за управление на книжен фонд и може да бъде доразвит в пълнофункционална библиотечна система според нуждите на конкретна организация.

# Използвани литературни източници и Уеб сайтове

1. [Официална документация на StyleCop Analyzers](https://github.com/DotNetAnalyzers/StyleCopAnalyzers)
2. [Unit тестване в .NET чрез xUnit](https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-csharp-with-xunit)
3. [Microsoft.AspNetCore.Identity.EntityFrameworkCore](https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-9.0&tabs=visual-studio)
4. [QuestPDF](https://www.questpdf.com/quick-start.html)
5. [SendGrid](https://www.twilio.com/docs/sendgrid)
6. [FluentAssertions](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

[Фигура № 1: Вход 11](#_Toc195018747)

[Фигура № 2: Регистрация 12](#_Toc195018748)

[Фигура № 3: Забравена парола 12](#_Toc195018749)

[Фигура № 4: Начална страница 13](#_Toc195018750)

[Фигура № 5: Книжен инвентар 13](#_Toc195018751)

[Фигура № 6: Добавяне на нова книга 14](#_Toc195018752)

[Фигура № 7: Наематели 14](#_Toc195018753)

[Фигура № 8: Управление на наеми 15](#_Toc195018754)

[Фигура № 9: Създаване на нов наем 15](#_Toc195018755)

[Фигура № 10: Връщане на книга 16](#_Toc195018756)

[Фигура № 11: Смяна на парола 16](#_Toc195018757)

[Фигура № 12: Генериране на PDF отчет 17](#_Toc195018758)

[Фигура № 13: Диаграма на случаите за употреба (Use Case) 19](#_Toc195018759)

[Фигура № 14: Диаграма на дейността №1 (Activity diagram) 20](#_Toc195018760)

[Фигура № 15: Диаграма на дейността №2 (Activity diagram) 21](#_Toc195018761)

[Фигура № 16: Диаграма на дейността №3 (Activity diagram) 21](#_Toc195018762)

[Фигура № 17: Диаграма на дейността №4 (Activity diagram) 22](#_Toc195018763)

[Фигура № 18: Диаграма на отношенията между моделите (Entity Relationship) 23](#_Toc195018764)

[Фигура № 19: Диаграма на отношенията между главните проекти 26](#_Toc195018765)

[Фигура № 20: Диаграма на отношенията между проектите и техните зависимости 28](#_Toc195018766)

[Фигура № 21: Диаграма на слоя от данни 29](#_Toc195018767)

[Фигура № 22: Диаграма на бизнес слоя 31](#_Toc195018768)

[Фигура № 23: Диаграма на презентационния слой 32](#_Toc195018769)