EstatePilot

# Даниел Николов 0MI0600410

Жан Георгиев 8MI0600398

# ESTATE PILOT

Изисквания към EstatePilot:

1. Достъпът до системата трябва да може да се осъществява през браузър или чрез мобилно приложение за iOS и Android.

2. Системата поддържа различни роли потребители, всяка от които използва специфични нейни функционалности:

a) Администратори – управляват съдържанието, следят за фалшиви обяви, дублиращи се обяви, некоректно публикувани обяви, коментари неотговарящи на правилата на системата и други.

b) Брокери – публикуват и скриват обяви, потвърждават часове за оглед на имота.

c) Физически лица – преглеждат обяви, записват часове за оглед, оценяват и публикуват обяви, публикуват обяви срещу индивидуална такса на обява.

d) Нерегистрирани потребители - могат да търсят и разглеждат обяви без регистрация.

3. За публикуване на обява са необходими следните данни:

a) Тип на обявата – продава / отдава под наем

b) Тип на имота – апартамент / офис / гараж c) Описание на имота

d) Локация e) Цена

f) Тип строителство (тухла, панел и др.)

g) Квадратура

h) Снимки

i) Контактна информация

j) Възможни часове за оглед и времетраене (достъпно само за брокери и по желание)

4. Системата визуализира обявите върху карта чрез интеграция с Google Maps, като може да се наложи интеграция и с други популярни услуги за географски карти като Apple Maps, Open Street Map и т.н.

5. Потребителите се регистрират в системата с потребителско име, имейл и парола.

6. За да може регистриран потребител да се регистрира като брокер, той трябва да заплати годишен абонамент чрез предоставените от системата методи и да предостави чрез специална форма документи удостоверяващи дейността му. Администратор има право да ги одобри или да отхвърли чрез специален екран – опашка в административната секция на системата. След одобрение системата автоматично променя статуса на потребителя на брокер.

7. Физическо лице може да публикува обяви, като преди да публикува обявата следва да заплати еднократна такса за всяка публикувана обява.

8. Системата предоставя плащания с карта, PayPal и еPay.

9. Всички финансови транзакции трябва да са защитени срещу измами и да бъдат обработвани надеждно.

10. Брокери могат да одобряват или отказват часове за оглед, като системата информира заявителя чрез имейл за резултата и при евентуално одобрение от брокера, блокира часа за запазване от други потребители.

11. Администратор може да блокира достъпа на брокери или физически лица в случай на установяване на нерегламентирано използване на системата.

12. Всички типове потребители могат безплатно да търсят по квартал, тип на имота, квадратура, цена, тип строителство.

13. Физическо лице може да запазва свободни часове за оглед на обяви публикувани от брокер, които подлежат на одобрение от брокера публикувал обявата.

14. Физически лица могат да оставят коментари върху обяви, както и да поставят оценка от едно до пет за имота, след като мине часа и датата за оглед.

15. Потребителите могат да поставят оценка от 1 до 5 на всеки профил на брокер в системата, за чиято обява са били на оглед.

16. Брокери и физически лица могат да редактират публикуваните от тях обяви, за което системата пази история.

17. Брокери и физически лица могат да архивират обяви, след успешна имотна сделка.

18. Да се предвиди интеграция на чат-бот чрез подходящ голям езиков модел, който да подпомага потребителите, например да препоръчва обяви, райони, и др.

19. Администраторите могат да преглеждат списъка с потребителите предоставили документи и заплатили такса за регистрация като брокери.

20. Администраторите могат да премахват обяви или да трият коментари в случай на нарушение на правилата на системата.

21. Администраторите имат право да блокират потребители поради нарушения на правилата на системата.

22. Системата поддържа кредитен калкулатор, който включва актуална информация за актуалните кредитни условия предлагани от банки – партньори. Извличането на данните става в реално време чрез интеграции със съответните банкови системи.

23. Регистрираните потребители могат да изискат справка за имота чрез директна интеграция със системата на Имотен Регистър с който да удостоверят собствеността върху разглеждания имот, като при поискване заплатят такса.

24. Системата трябва да поддържа възможност за скалиране при високо натоварване през месец август и септември, които са пикови за пазара на имоти. От друга страна системата трябва да позволява скалиране надолу през зимните месеци, характерни с малкия брой сделки с цел оптимизация на разходите.

25. Системата трябва да е 99.9999% налична.

26. Системата трябва да защитава личните данни на потребителите и да е напълно съвместима с изискванията на GDPR. Всяка обява трябва да показва само необходимата контактна информация, без да излага на риск личните данни на потребителя.

# СОФТУЕРНА АРХИТЕКТУРА:

**1. Достъпът до системата трябва да може да се осъществява през браузър или чрез мобилно приложение за iOS и Android.**

За да поддържаме достъп до системата през браузър или чрез мобилно приложение за iOS и Android, е необходимо да използваме Client-Server стил на архитектура. Освен това приложението трябва да отговаря на политика на публикуване на приложения в Google Play и App Store.

**2. Роли: администратори, брокери, физически лица, нерегистрирани**

Системата поддържа различни типове потребители, като администратори, брокери, физически лица и нерегистрирани, всеки с различни нива на достъп и функционалности. Реализацията се основава на архитектурен стил Model-View-Controller (MVC), който улеснява разделянето на логиката според потребителските роли.

**3. Интеграция с картови услуги (Google Maps, Apple Maps)**

Интеграцията с картови услуги (Google Maps, Apple Maps) се реализира чрез Service-Oriented Architecture, в която специализиран микросервис (напр. “MapService”) управлява всички заявки, свързани с локации и карти. За да се постигне гъвкавост и независимост от конкретен доставчик, се използва Adapter/Wrapper слой, който абстрахира комуникацията с външните API-та. Това позволява лесна поддръжка, смяна на доставчици и спазване на политиките за ползване на картовите услуги.

**4. Регистрация с имейл и парола**

Регистрацията с имейл и парола се реализира чрез Authentication модул, който обработва създаването на потребителски акаунти. Паролите се съхраняват с шифроване (hashing) чрез алгоритми като bcrypt или Argon2, за да се гарантира сигурността на данните. Модулът също така поддържа валидация на имейл, проверка за уникалност и изпращане на потвърждение при регистрация.

**5. Плащания с карта, PayPal и ePay**

Плащанията с карта, PayPal и ePay се реализират чрез Payment Service, който интегрира външни платежни доставчици чрез защитени API-та. Системата трябва да бъде PCI-DSS съвместима, което означава, че не съхранява чувствителни картови данни и използва токенизация или посредници. Това осигурява сигурност при трансакции и лесна поддръжка на множество методи на плащане.

**6. Защита на финансовите транзакции**

Защитата на финансовите транзакции се осигурява чрез TLS криптиране на всички данни в движение, което предпазва от прихващане и подслушване. Използват се анти-фрод механизми, като лимити, поведенчески анализ и проверки по IP или устройство. Всички транзакции се логват детайлно за целите на проследимост, одит и разследване при съмнения за злоупотреба.

**7. Запазване и одобрение на часове за оглед**

Запазването и одобрението на часове за оглед се управлява от Scheduling модул, който координира заявките между потребители и налични слотове. За избягване на двойни резервации се прилага защита от конфликт на часове чрез заключване на времевите интервали. Успешните или променени резервации се потвърждават чрез Notification service (имейл, push известия).

**8. Блокиране на потребители от администратор**

Блокирането на потребители от администратор се реализира чрез User moderation модул, който позволява временно или постоянно ограничаване на достъпа. Всички действия по модериране се записват в логове за проследимост и отчетност.

**9. Търсене без регистрация**

Търсенето без регистрация се реализира чрез интегриран Search Engine (напр. ElasticSearch), който позволява бързо и мащабируемо индексиране на обяви. Потребителите могат да използват филтри по критерии като категория, цена, локация и др. без да е необходим вход в системата. Това подобрява достъпността и улеснява първоначалното взаимодействие с платформата.

**10. Коментари и оценки от потребители**

Коментарите и оценките от потребители се реализират чрез Ratings & Comments модул, който позволява добавяне на мнения и звездни оценки към обяви или потребители. За поддържане на качеството на съдържанието се прилагат филтри за модерация, като автоматично откриване на неподходящ език и възможност за докладване.

**11. Кредитен калкулатор с банки**

Кредитният калкулатор използва актуална информация от различни банкови институции, което налага интеграция с множество външни банкови API-та. За целта се прилагат адаптери, които осигуряват унифицирана комуникация и позволяват синхронизация на данни в реално време. Системата трябва да гарантира висока достъпност и устойчивост, включително възможност за работа при временна липса на отговор от външна банка.

**12. Интеграция с Имотен Регистър**

Интеграцията с Имотния Регистър изисква връзка с публични или частни API-та, чрез които се осъществява достъп до собственост и правен статут на имоти. За целта се използват адаптери, които абстрахират различните формати и технологии на външните системи. Поради рядката актуализация на данните, се прилага кеширане с цел бързо зареждане и по-добро потребителско изживяване. Допълнително се внедряват механизми за устойчивост – обработка на таймаути, неуспешни заявки и автоматично повторение при грешки.

**13. Автоматично скалиране според натоварване**

За да осигурим висока производителност и достъпност на системата при променливо потребителско натоварване, се използват автоматизирани механизми за скалиране чрез Cloud ресурси. Автоматичното скалиране се управлява чрез инструменти, които следят натоварването на системата и предприемат действия в реално време.

**14. Висока наличност (99.9999%)**

За да се предотврати претоварване при високо натоварване, се внедрява приоритетна опашка, която осигурява първостепенна обработка на ключови заявки и задачи. Това позволява по-важните операции да бъдат изпълнявани с предимство, докато второстепенните чакат своя ред, без да блокират системата. Освен това, системата трябва да поддържа няколко инстанции на критични компоненти, като микросервиси за плащания, търсене и автентикация, за да се гарантира висока достъпност и устойчивост при отказ или повишено търсене.

**15. GDPR съвместимост и защита на лични данни**

Системата трябва да бъде напълно съвместима с Общия регламент за защита на личните данни (GDPR), като гарантира прозрачност, контрол и сигурност при обработка на лична информация. За осигуряване на защита се прилагат технически и организационни мерки, като криптиране на данни, контрол на достъпа, псевдонимизация, периодично логване и мониторинг, както и договори за обработка с всички външни доставчици на услуги.

ОПИСАНИЕ НА МОДУЛИ

1) AUTHETICATION MODUL - има за цел да обработва създаването на потребителски акаунти. Модула поддържа валидация на имейл, проверка за уникалност и изпращане на потвърждение на данните. Във него има 4 подмодула. Единия е имплементация на MVC архитектура за мениджър на роли. Другия е модул за шифроване на пароли чрез алгоритми. Има модул за валидация на имейл и парола, както и за двуфакторна верификация.

2) SEARCH ENGINE - Този модул позволява бързо и мащабируемо индексиране на обяви. Има 1 подмодул в него, който отново е имплементация на MVC архитектурата.

3) USER MODERATION - Модулът отговаря за временно или постоянно ограничаване на достъпа, както и за одобряване на акаунти на брокери. Има 2 подмодула в него, всеки изпълняващ съотвено ролите си. Освен това, модулът е съобразен с изискванията на GDPR и гарантира защита на личните данни при изтриване. При искане на потребителя, той може да бъде напълно забравен.

4) RATINGS AND COMMENTS - Модулът позволява добавяне на мнения и звездни оценки към обяви или потребители. Има 4 подмодула в него. Първият извършва проверка за нецензурни коментари. Втория съдържа информацията за създателя на коментара. Третия се използва за употребата на оценките. Последния се грижи за съдържанието на самите коментари.

5) BANK ADAPTER - Модулът менажира заявките които са изпратени или приети. Има 2 подмодула вътре. Първия се грижи за изпратените, а втория за приетите отговори.

6) PAYMENT SERVICE - Moдулът интрегрира външни платежни доставчици чрез API-та. В него има 3 подмодула. Единия се грижи за шифроването на чувствителните картови данни, другия е за 2-факторна верификация, а третият менажира заявки за плащане.

7) CREDIT CALCULATOR - Модул, който използва MVC архитектура за обработка на заявки към външни API-та. Модулът приема заявките, комуникира с външните услуги и визуализира получените резултати. Адаптерът осигурява гъвкавост и слаба свързаност, позволявайки лесна интеграция с различни банки.

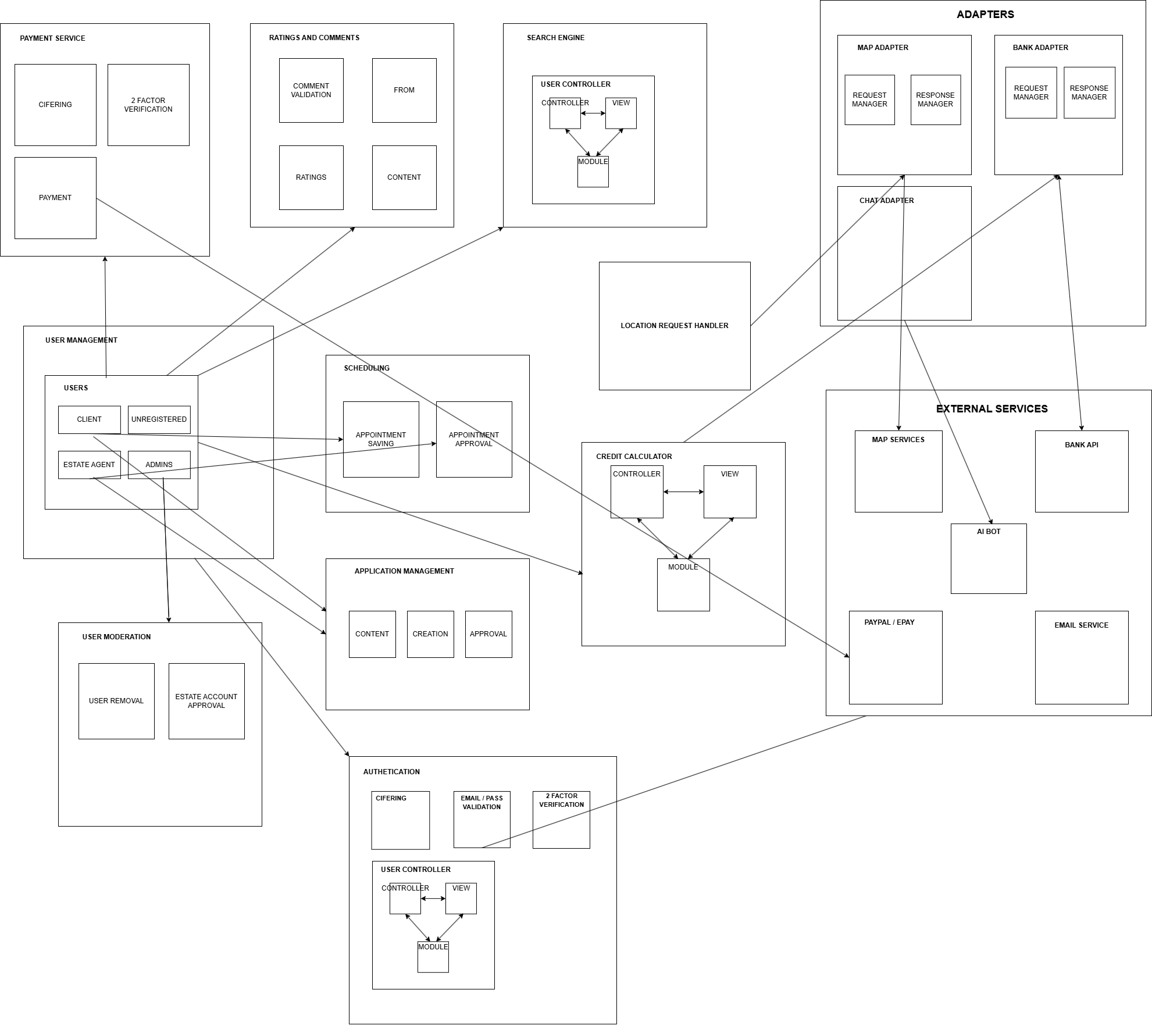
8) USER MANAGEMENT - Модулът съхранява информация за различните видове потребители на системата.

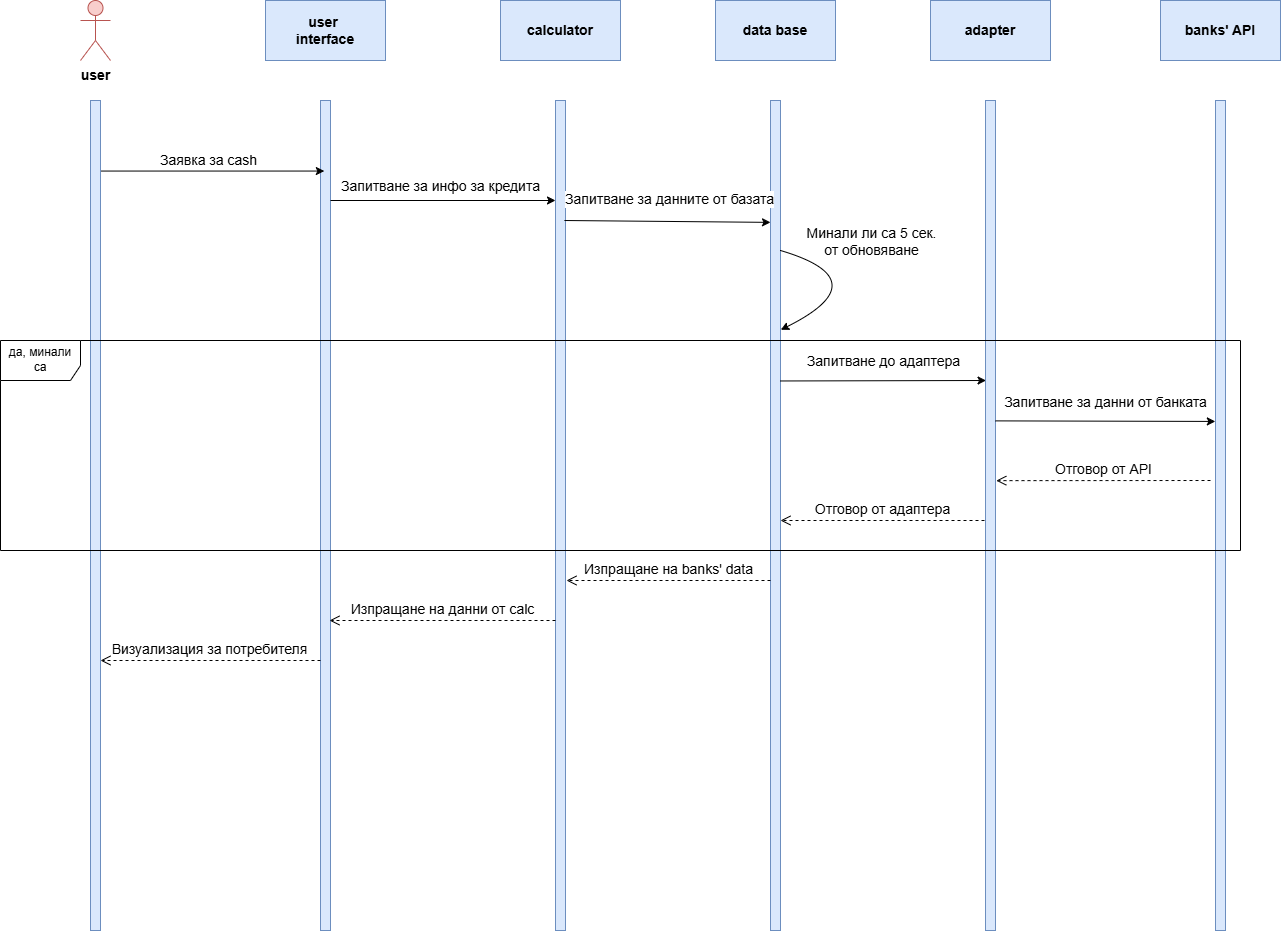
9) APPLICATION MANAGEMENT - Модулът менажира създаването на обяви, тяхното съдържание и одобрението им. Има 3 подмодула като всеки се грижи за съответната роля.

10) SCHEDULING - Модулът служи за запазване и одобряване на заявки за часове за оглед. В него има 2 подмодула като всеки изпълнява съответната роля.

11) AI BOT - Системата включва интелигентен AI бот, който може да помага на потребителите – например да им препоръчва подходящи обяви според търсенето им, да обяснява термини или да им отговаря на въпроси за имоти, райони и процеси в платформата.

12) CHAT ADAPTER - За да работи AI ботът, имаме адаптер, който осъществява връзката между системата и голям езиков модел. Адаптерът превежда съобщенията на потребителя в подходящ формат и връща отговорите на разбираем език. Въпросите се филтрират така, че да са свързани с това, което системата може да прави.





**Описание на Изгледа:**

Потребителят прави заявка през интерфейса на системата – иска информация за кредит („заявка за cash“). Интерфейсът подава тази заявка към калкулатора, който се грижи за обработката на такава информация.

Калкулаторът прави запитване към базата данни, за да види дали вече има информация за кредитните условия и дали не е твърде стара – проверява дали са минали повече от 5 секунди от последното обновяване.

Ако е минало време (да, минали са 5 сек.), базата казва, че трябва да се направи ново обновяване на данните. В такъв случай:

1. Базата изпраща запитване до адаптера, който служи като посредник между системата и външните банки.
2. Адаптерът прави реално API запитване до банковата система, за да вземе най-актуалните данни.
3. Банковият API връща информацията към адаптера.
4. Адаптерът връща отговора обратно към базата данни.
5. След това базата праща тези „banks’ data“ към калкулатора.

Калкулаторът обработва данните и ги изпраща обратно към потребителския интерфейс, който визуализира информацията пред потребителя.

**Описание на Обкръжението:**

**User** – Човекът, който използва приложението и иска да получи информация за кредит. Той инициира процеса чрез потребителския интерфейс.

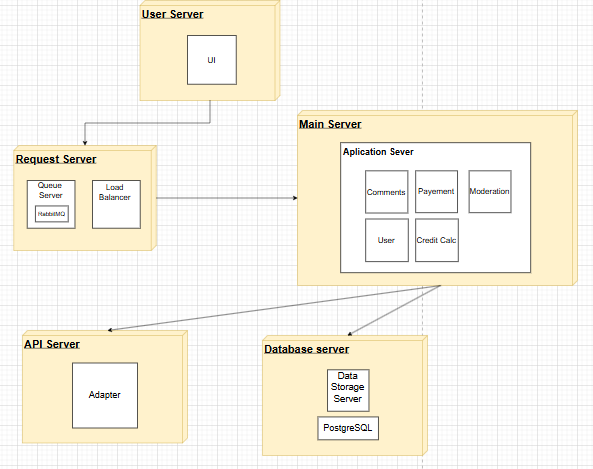
**User Interface** – Това е мястото, където потребителят прави заявка и получава визуален отговор. Интерфейсът приема вход от потребителя и показва резултатите след обработка.

**Calculator**  – Компонент от системата, който съдържа бизнес логиката. Той отговаря за обработката на заявката и за определяне дали трябва да се направи ново запитване към външните източници.

**Database** – Служи за съхранение на вече получени данни за кредити. Преди да се прави външно запитване, системата проверява тук дали вече няма налична и актуална информация, за да се избегнат ненужни заявки.

**Adapter** – Действа като посредник между нашата система и външните банки. Превежда заявките от нашата система в подходящ формат за банките и обратно – отговора от тях в разбираем за user-a вид.

**Banks' API**  – Това е външната система на партньорските банки, която предоставя актуална информация за кредитните условия. Данните се извличат в реално време чрез нея.



## **Мотивация за избор на диаграма на внедряването**

В рамките на архитектурната документация на системата Estate Pilot, диаграмата на внедряването заема важно място, тъй като показва къде и как точно се разполагат основните части на системата – независимо дали става въпрос за физически сървъри, виртуални машини, облачни услуги или устройства на потребителите.

Този тип диаграма не само показва логическото разделение между модулите на системата, но и дава ясна представа за тяхното реално разположение, как комуникират помежду си, през какви мрежи и протоколи, и от какви ресурси зависят.

Диаграмата на внедряването е много подходяща за Estate Pilot, защото показва как работи цялата система – от приложенията, които ползват хората, през сървърите и базите данни, до връзките с външни услуги като плащания, карти и кредитна информация. Така става ясно кои технологии как си взаимодействат, как се разпределя натоварването и как е осигурена защитата на системата.

Освен това, тази диаграма ни помага да визуализираме как системата ще се държи в реална среда – например как уеб приложението комуникира със сървъра, как се извършват плащания, как се пазят лични данни, и как са организирани резервните системи. Можем да покажем кои услуги са отделени в самостоятелни сървъри или контейнери, какви мрежови връзки има между тях и кои компоненти са критични за работата на платформата.

Важен плюс на този тип диаграма е възможността да представим и стратегията за мащабиране – както при натоварване (напр. през август и септември, когато търсенето на имоти е високо), така и при намалена активност (през зимните месеци). Това ни дава основа за оптимизация на ресурсите и планиране на инфраструктурата.

В заключение, изборът на диаграма на внедряването е логично и ефективно решение за представяне на цялостната архитектура на Estate Pilot. Тя дава ясен и структуриран поглед към това как всичко е организирано – както от софтуерна, така и от хардуерна гледна точка – и помага да се обяснят важни решения, свързани с надеждност, сигурност и мащабируемост.

### **User Server**

Това е клиентската част на системата – може да бъде уеб браузър или мобилно приложение. Тук потребителите взаимодействат с платформата – разглеждат обяви, търсят имоти, резервират огледи, пишат коментари и др.

Всички действия от потребителите се изпращат под формата на заявки към Request server, който ги обработва и препраща към основната логика.

**Request Server**

Този сървър отговаря за приемане и правилно разпределяне на потребителските заявки.

Съдържа Queue Server и Load Balancer:

**RabbitMQ** служи за буфериране – когато системата е претоварена, той съхранява заявките и ги обработва по ред.

**Load Balancer** добавя нови ресурси при високи натоварвания и решава кой от компонентите на Main Server да поеме заявката, така че да се постигне баланс и да се избегне претоварване.След това заявките се насочват към Main Server, където се обработват от основната логика на приложението**.**

**Main Server**

Това е сърцето на системата. Тук се намира цялата бизнес логика – взимат се решения, валидират се действията на потребителите и се обработват заявките.

Съдържа Application Server с няколко микросервиса:

**User Service** – регистрация, логин, управление на роли и пароли.

**Comments** – система за оценки, ревюта и филтриране на нецензурно съдържание.

**Payment** – отговаря за плащания, токенизация, 2FA проверки.

**Moderation** – одобрение или блокиране на потребители / брокери.

**Credit Calculator** – изчисляване на кредити, комуникация с банкови API-та.

**API Server**

Този сървър комуникира с външни услуги, от които системата зависи:

* Картови услуги
* Банкови системи
* Имотен център

Той осигурява нужната информация на Main Server – например, при показване на карта или при калкулиране на кредитна оферта.

**Database Server**

Това е мястото, където се съхранява цялата важна информация в системата. Сървърът използва PostgreSQL и отговаря за надеждното съхранение на: потребителски профили, обяви за имоти, резервации и история на огледи, данни за плащания, коментари и рейтинги.

Всички останали части на системата разчитат на тази база, за да получават и записват нужната информация бързо и сигурно.