Health & fitness tracker

Автори:

Антония Игнатова, Васил Ганев,

Светослав Татерски, Симеон Цеков

Дата: 24.10.2023

Въведение

Този документ представя архитектурния проект на мобилното приложение Health & fitness tracker. Целта му е да покаже какви архитектурни решения са взети и какъв ще е процесът на разработка. Изяснява се какви данни и компоненти ще се използват, какви процеси ще се извършват и как ще се реализират атрибутите на качество.

Участници са: потребители и софтуерни разработчици.

Предназначение

Обхват

Документът обхваща същинската част на архитектурния проект. Определя се архитектурата на системата, включително структура, компоненти, връзки, интерфейси, архитектурни шаблони и технологиите за разработка. В него присъстват и имплементационни детайли като диаграми и структури на база данни, които помагат на разработчиците да изградят системата.

Термини

Ø API (Application Programming Interface) - набор от протоколи и правила, които позволяват на различни софтуерни приложения или компоненти да комуникират помежду си.

Ø Модел - представлява сърцевината на приложението и съдържа данните, бизнес логиката и правилата, свързани с обработката на данни.

Ø Вюмодел - представлява структура или клас, който се използва, за да предостави подходящо форматирани данни и състояние на интерфейсните компоненти.

Ø REST (Representational State Transfer) - архитектурен стил за разработка на уеб приложения, който се основава на принципи за изграждане на лесно разширяеми, мащабируеми и преизползваеми системи.

Ø JSON (JavaScript Object Notation) - опростен формат за обмен на данни, удобен за работа както за хората, така и за компютрите.

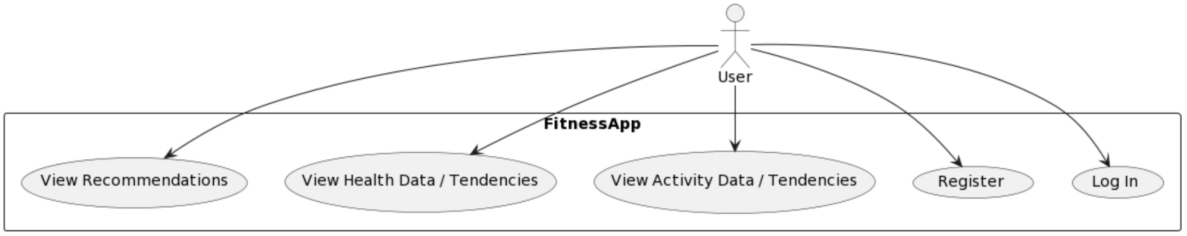
Източници

drawIO - https://app.diagrams.net/
plantUML - https://plantuml.com/
Wikipedia

GitHub repo - https://github.com/SimeonTsekov/TU_ST

Архитектурен обзор

Use-case изглед

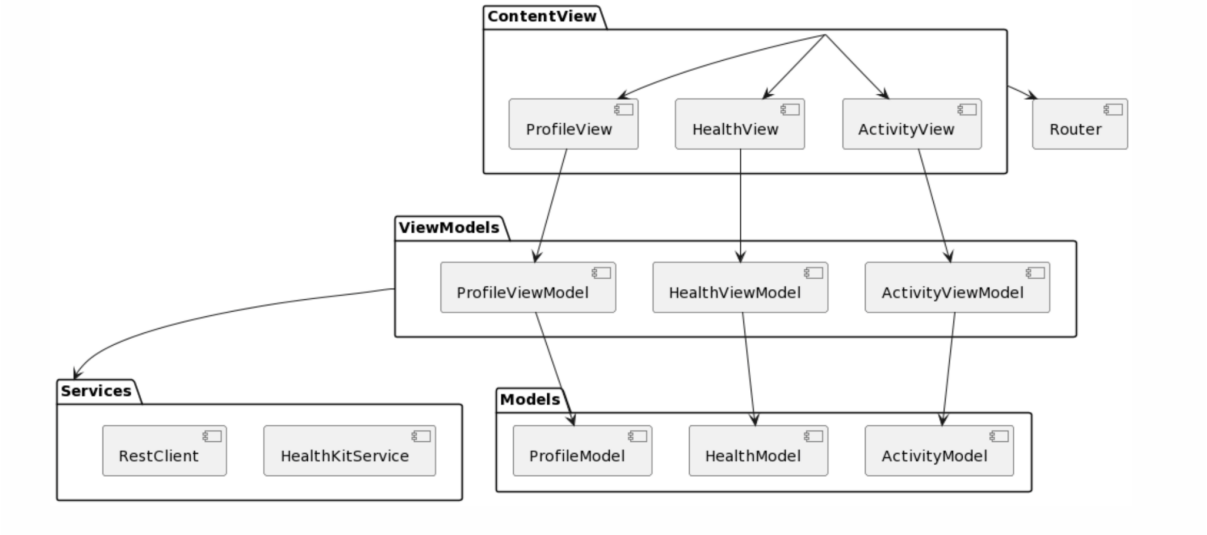


Приложението ще съдържа само един актьор - потребител. Потребителят ще може да:

- използва фитнес приложението, за да влезе в своя акаунт (UC1);
- да създаде нов акаунт (UC2);
- да преглежда данни и тенденции за активността си (UC3);
- да преглежда здравни данни и тенденции (UC4);

- да получава препоръки, генерирани на база данните за съответния потребител (UC5).

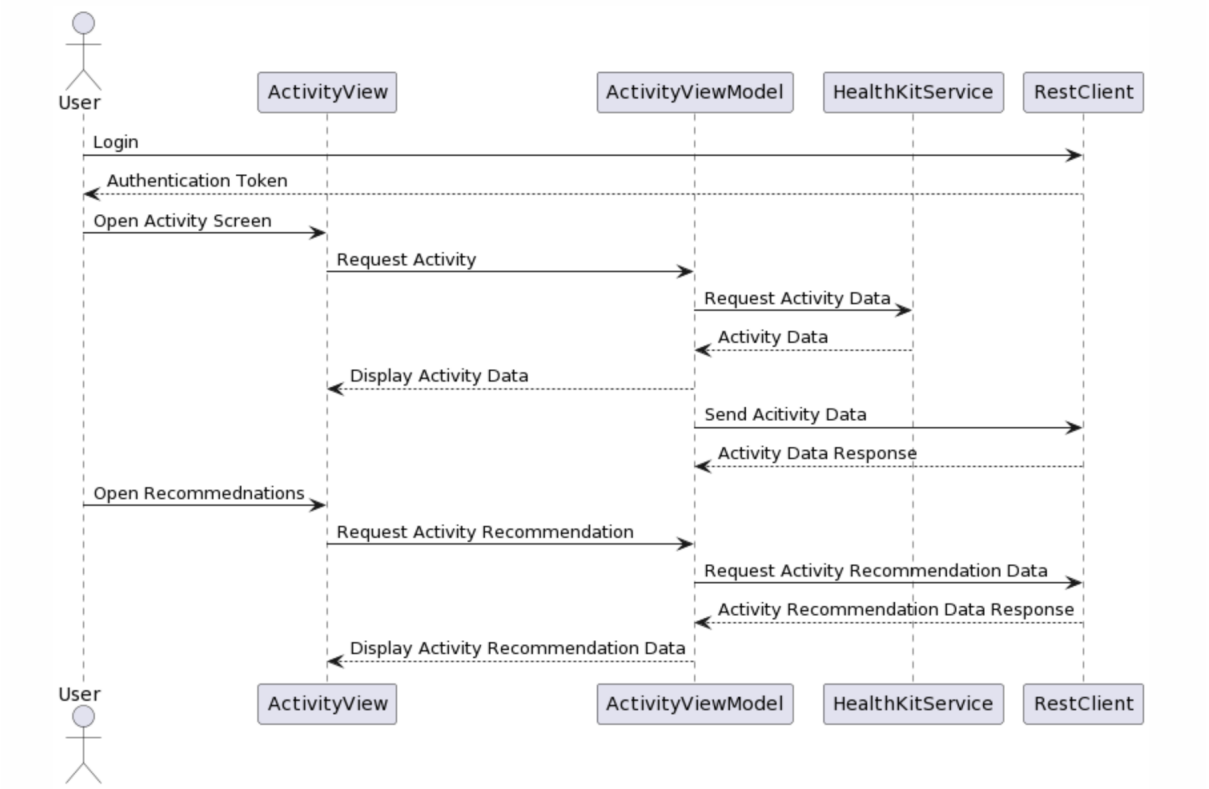
Логически изглед



Диаграмата на класовете представя груба структур на приложението за фитнес и включва:

- ContentView, съдържащо три изгледа - ActivityView, HealthView, ProfileView;
- съответните им модели и вюмодели;
- HealthKitService, осъществяващ връзката с HealthKit;
- REST клиент за комуникация с бекенда.

Процесен изглед

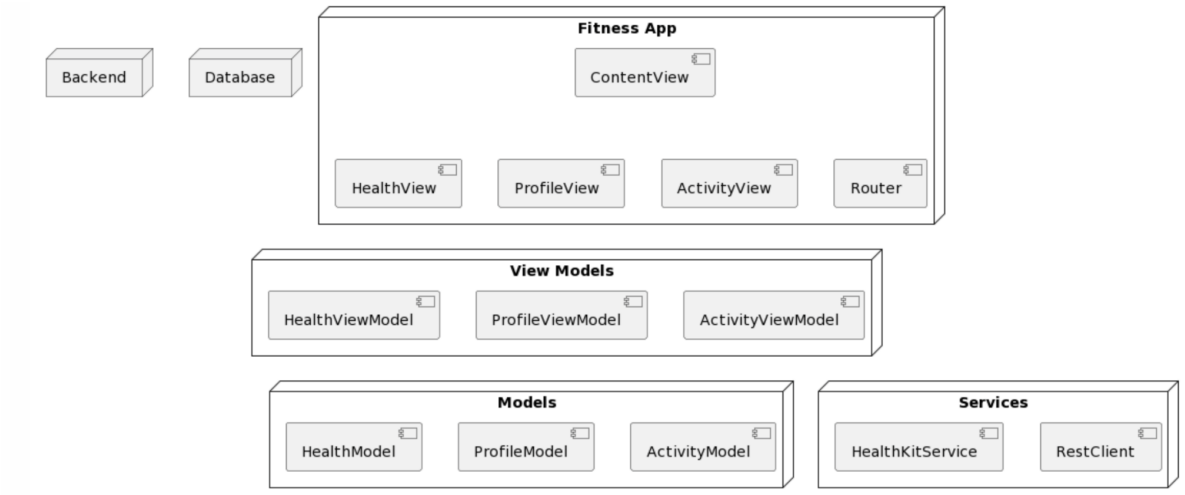


Разработената диаграма на последователността представлява визуален обзор на основните действия и взаимодействия в приложението. Потребителят (User) започва, като се аутентикира чрез вход в системата и получава уникален токен за удостоверение. След успешния вход, той отваря екрана за активност (ActivityView), който се обръща към съответния вюмодел (ActivityViewModel). Този вюмодел осъществява комуникация със сървиса HealthKitService, за да достъпи актуални данни за активност. Получените данни се визуализират и представят на потребителя в ActivityView.

Потребителят има и възможността да достъпи препоръки за своята активност, на база заявка към бекенда, посредством REST клиента. Получената информация се визуализира в екрана за активност.

По аналогичен начин работи и здравният екра, като единствената разлика ще бъде типът данни, които се визуализират, както и начинът, по който ще се визуализират.

Компонентен изглед



Този изглед предоставя цялостна представа за архитектурата на приложението:

- моделите представляват обектите от данни, с които приложението и сървърът ще работят;
- вюмоделите предоставят бизнес логика на ниво use-case, т.е. всеки от съответните екрани;
- сървизите ще предоставят връзка с външни модули като сървъра и HealthKit;
- екраните ще представляват потребителския интерфейс, който ще визуализира наличните данни и ще предоставя интерактивност;
- сървърът ще събира и обработва данните и ще ги записва, както и ще генерира нови данни
- базата данни ще служи за съхранение на всички данни.

Изглед на данните

1. User таблица

Основната таблица, която съхранява информация за всеки индивидуален потребител.

- userID - Уникален идентификатор, който позволява връзка с останалите таблици.
- username - Уникален потребителско име за всяко лице.
- email - Електронна поща на потребителя.
- password - Хеширана парола за сигурност.

- age - Възраст на потребителя.
- height - Височина на потребителя в см.

2. Health Data таблица

Съхранява различни здравни записи за всеки потребител.

- userID - Уникален идентификатор, свързващ с User таблицата.
- bodyMass - Тегло на тялото в кг.
- BMI - Индекс на телесната маса.
- bodyFat - Процент на телесната мазнина.
- leanBodyMass - Чиста телесна маса без мазнини.
- sleepAnalysis - Анализ на качеството и продължителността на съня.

3. Activity Data таблица

Съхранява различни записи за физическата активност на потребителя.

- userID - Уникален идентификатор, свързващ с User таблицата.
- workouts - Видове тренировки и продължителност.
- dailySteps - Брой стъпки на ден.
- dailyDistance - Изминато разстояние на ден в км.
- dailyEnergyBurned - Изгорени калории на ден.

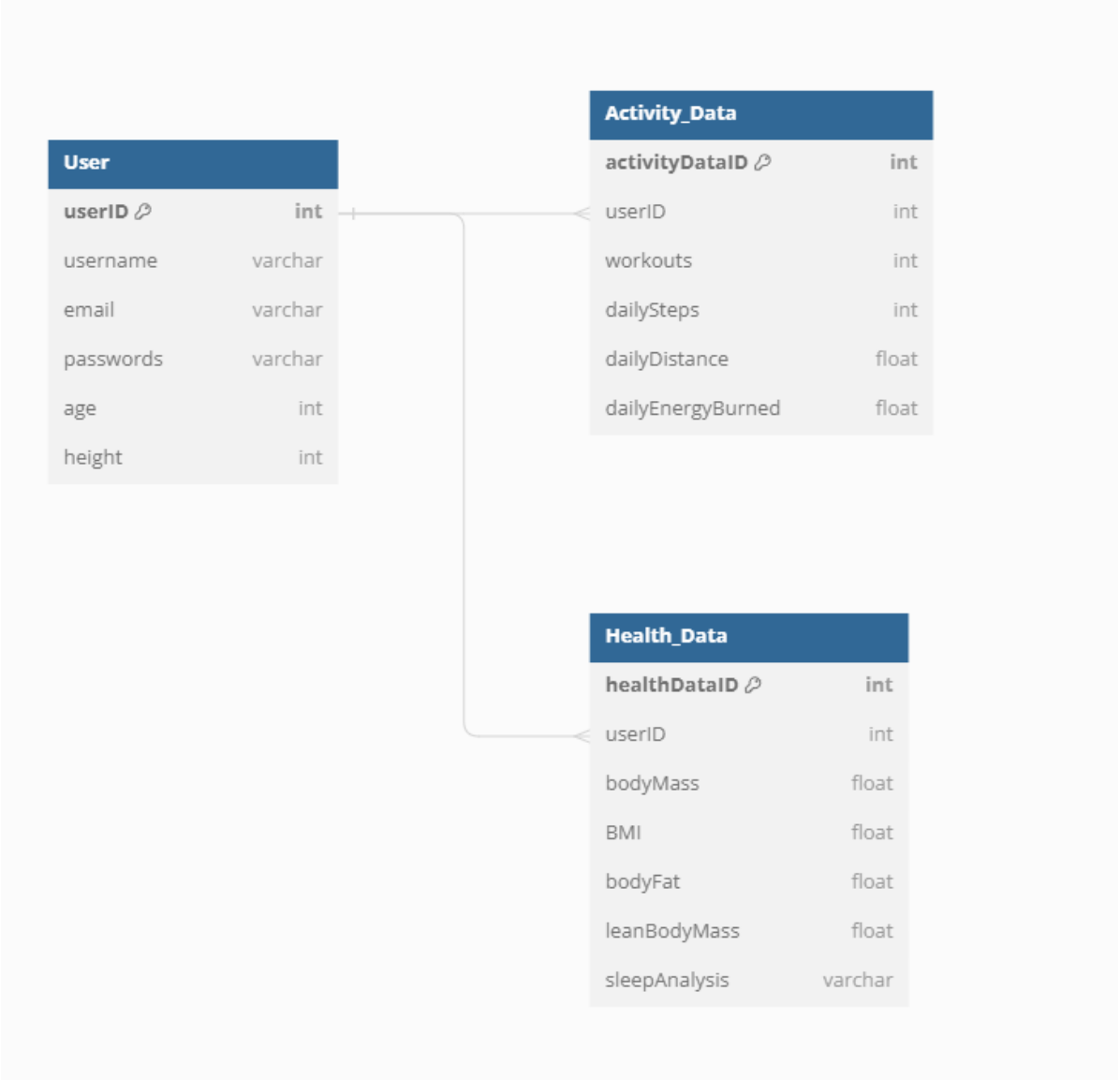
4. Recommendation Model 1

Моделът за препоръки може да взема информация от горните таблици и да предлага конкретни тренировки или диети.

Обобщение на ключовите моменти:

- Четири основни таблици: User, Health Data, Activity Data и модел за препоръки.
- Връзката между User таблицата и останалите таблици е 1:много.
- Всеки потребител има уникален идентификатор, потребителско име и електронна поща.

- Здравните данни включват информация за телесната маса, мазнините, чистата телесна маса и съня.
- Данни за активността предоставят информация за тренировките, стъпките, изминатото разстояние и изгорените калории.



Изглед на внедряването:

Хардуерни ресурси:

1. Сървъри
   - Основен сървър за база данни (MySQL), която съхранява информацията за потребители, здравословни данни и активност.

- Уеб сървър - Отговаря за обслужване на заявките от потребителите и предоставяне на интерфейса на приложението.

2. Потребителски устройства

- Мобилни телефони, чрез които потребителите имат достъп до системата и може да участват в различни активности или да проследяват своята информация.

Комуникация

- Интернет - Приложението е предназначено за широк достъп, комуникацията между сървърите и потребителските устройства ще се осъществява чрез интернет.

Изглед на имплементацията:

1. Архитектурна концепция
   - Модел на слоеве - Приложението ще следва модел на разделение на слоеве, което подпомага отделянето на функционалностите и гарантира модуларност, лесна поддръжка и разширяемост.

2. Слоеве

Презентационен слой

- Предназначение - Отговаря за интерфейса и потребителската интеракция.
- Имплементация и употреба - Ще използва Swift за разработка на потребителския интерфейс за iOS устройства.

Слой за логика на приложението (Back-end)

- Предназначение - Обработва логиката на приложението, управлява бизнес правилата и комуникира с базата данни.

- Имплементация и употреба - В този слой ще се използва .NET Core Web API, който обработва заявките от презентационния слой и връща данни към него.

Слой за данни

- Предназначение - Отговаря за съхранението и достъпа до данни.
- Имплементация и употреба - MySQL е избраната база данни, която ще управлява информацията за потребители, здравни данни и активност.

3. Комуникация между слоевете:
    - Презентационният слой ще комуникира със слоя за логика на приложението чрез RESTful API, предоставен от .NET Core Web API, за да изпълни операции като извличане, обновяване, добавяне на данни.
    - .NET Core Web API ще използва драйвери и библиотеки за връзка с MySQL базата данни за изпълнение на операции с данните.

4. Безопасност:
    - Комуникацията между слоевете, особено между презентационния слой и .NET Core Web API, ще бъде криптирана чрез SSL/TLS.
    - Всички данни, свързани с пароли на потребителите, ще бъдат хеширани в базата данни.

Нефункционални изисквания

1. Достъпност

Достъпността в контекста на софтуерната архитектура се отнася до способността на системата да бъде достъпна и използвана от различни потребители, включително лица с различни степени на физически или когнитивни ограничения. Достъпността е важна не само за потребителите с увреждания,

но и за по-широката аудитория, тъй като подобрява цялостната използваемост на софтуерното приложение.

a. Навигация и интерфейс – Потребителският интерфейс ще е направен така, че да бъде лесен за използване, както за потребители с увреждания, така и за потребители без такива.
b. Алтернативни медийни форми – Дизайнът ще бъде адаптивен и ще позволява на приложението да се съобразява с различни размери екрани. Това подобрява използваемостта на мобилни телефони или таблети.

2. Разширяемост

Разширяемостта се отнася до способността на системата да бъде лесно променяна или разширявана с добавяне на нови функции, модули или компоненти. Разширяемите системи могат да се адаптират към променящи се изисквания и нужди на потребителите без значителни промени в основната архитектура.

a. Модулна архитектура – Приложението ще бъде разделено на независими модули, които ще могат да бъдат лесно разширявани. Това ще позволи както и на нас така и на бъдещите разработчици да добавят нови модули без да променят основната архитектура.
b. API и документация – Ще бъде представена документацията за API и интерфейси, които ще улесняват бъдещите разработчици с разширяването на приложението.

3. Производителност

Производителността е важен атрибут на качеството в софтуерната архитектура и се отнася до способността на системата да изпълнява своите функции бързо и ефективно, без да изпитва натоварване на изпълнението при увеличение на обема на данни или заявките.

a. Оптимизиран код – Приложението ще бъде направено с оптимизирани алгоритми и структурни данни. Ще бъдат избегнати излишни операции и обработки, които могат да забавят изпълнението.
b. Минимални изисквания към ресурсите – Приложението ще бъде написано с оглед минимално използване на ресурси като процесорно време, памет и енергия. За мобилните устройства това е особено важно, защото там ресурсите са ограничени.
c. Асинхронна обработка – Където е възможно, ще бъде използвана асинхронна обработка за да се избегне блокиране на потребителският интерфейс по време на по-дълги операции.

4. Сигурност

Сигурността е от решаващо значение за проекта, тъй като приложението съхранява лични и здравни данни на потребителите.

a. Автентикация и удостоверяване – Ще бъде осигурена надеждна и силна система за автентикация, която ще валидира потребителя. Ще бъдат използвани сигурни методи като хеширане на пароли, за да бъдат защитени потребителските акаунти.

5. Възможност за тестване

Възможността за тестване на приложението е от съществено значение, за да гарантираме, че то работи коректно, сигурно и отговаря на очакванията на потребителите.

a. Тестове на реални потребители – Извършване на тестове с реални потребители, които да дават обратна връзка. Така може да проверим как приложението работи в реална среда за употреба.

6. Интероперабилност

Интероперабилността се отнася до способността на приложението да е съвместимо с различни устройства,

платформи и системи. Това означава, че то трябва да може да обменя данни и взаимодейства с други приложения и устройства, за да предостави по-широка функционалност и удобство на потребителите.

a. Стандартизирани формати – Използване на стандартизирани формати за обмен на данни като JSON и XML с цел улеснение на обмяната на информация с други приложения и системи.
b. Отворен код и общностна поддръжка – Възможно е да публикуваме част от кода като отворен, за да може други разработчици да създават допълнения и интеграции с нашето приложение.

7. Използваемост

Използваемостта е ключов аспект, тъй като допринася за удовлетвореността на потребителите и успеха на приложението. Отнася до това колко лесно и удобно потребителите могат да взаимодействат с приложението и да постигат своите цели.

a. Потребителски интерфейс – Той ще бъде интуитивен и лесен за разбиране. Дизайнът ще бъде с разбираеми икони бутони и текст.
b. Навигация – Навигацията също ще бъде лесна и логична, защото потребителите трябва да могат да се ориентират по-лесно в приложението.