Software Architecture Document

Mobile application for improving self-discipline in gaming form (Gamify Your Life).

Kalinichev Hlib, Dgosan Volodymyr, Andriichenko Danyil

Version 1.0

Table of Contents

1. Introduction

1.1 Purpose

This document serves as a comprehensive guide to the architectural design and structure of our application, aiming to provide a clear understanding of the system's architecture for all stakeholders involved in the development, maintenance, and enhancement of the application. The primary purpose of this document is to outline the high-level software architecture, design decisions, and key components that constitute the Gamify Your Life application. By documenting the architecture, we aim to facilitate communication among development teams, project managers, and other stakeholders. This document also acts as a reference for developers, helping them comprehend the system's architecture and make informed decisions during the development life cycle.

1.2 Scope

The scope of this Software Architecture Document encompasses the architectural aspects of the Gamify Your Life mobile Android application. It includes an overview of the system's architecture, major components, their interactions, and the rationale behind architectural decisions. The document will cover both functional and non-functional requirements that impact the architecture, providing insights into how the application is structured and how it satisfies the specified requirements.

This document is intended for various stakeholders, including but not limited to developers, project managers, quality assurance teams, and system administrators. It will aid in the ongoing development, maintenance, and evolution of the Gamify Your Life application, ensuring a shared understanding of the software architecture across the project team.

2. Architectural Representation

This section provides a visual representation of the architectural design of the Gamify Your Life mobile Android application. The architecture is depicted through a set of diagrams and models, illustrating the key components, their relationships, and the flow of data and control within the system. Its primary purpose is to offer stakeholders a clear and concise overview of the system's architecture, focusing on key components, their interactions, and the overall flow of data and control within the application.

3. Architectural Goals and Constraints

3.1 Architectural Goals

3.1.1 Scalability

Goal: Design an architecture that can efficiently scale to accommodate a growing user base and additional features without compromising performance.

Rationale: As Gamify Your Life gains popularity and feature complexity, the architecture should support seamless scalability to maintain responsiveness and user satisfaction.

3.1.2 Maintainability

Goal: Foster a codebase that is easy to understand, modify, and extend, promoting efficient maintenance and future enhancements.

Rationale: A maintainable codebase accelerates the development cycle, facilitates bug resolution, and enables the incorporation of new features or updates with minimal disruption.

3.1.3 Performance Optimization

Goal: Optimize the application's performance to provide a responsive and smooth user experience, even under varying network conditions.

Rationale: Users should experience minimal latency and smooth interactions, contributing to a positive perception of the Gamify Your Life application.

3.1.4 User Experience (UX) Design

Goal: Implement an intuitive and visually appealing user interface that enhances user engagement and encourages consistent usage of the application.

Rationale: A well-designed UX contributes to user satisfaction, retention, and the overall effectiveness of the self-discipline improvement features.

3.1.5 Security and Data Privacy

Goal: Prioritize the security of user data and ensure compliance with relevant privacy regulations.

Rationale: Users should trust Gamify Your Life with their personal data, and robust security measures are essential to protect against potential threats and breaches.

3.2 Architectural Constraints

3.2.1 Mobile Platform Compatibility

Constraint: The application must be developed specifically for the Android platform.

Rationale: Gamify Your Life is designed as a native Android application to leverage platform-specific capabilities and provide a seamless user experience for Android users.

3.2.2 Budgetary Constraints

Constraint: Development efforts should adhere to the predefined budget for the project.

Rationale: Financial constraints necessitate efficient resource allocation and cost-effective development practices to ensure the project's economic viability.

3.2.3 Compliance with Third-Party APIs

Constraint: Any third-party APIs used in the development of Gamify Your Life must adhere to licensing agreements and terms of service.

Rationale: Compliance with third-party API terms is essential to avoid legal issues and ensure a stable and supported integration.

3.2.4 Network Connectivity

Constraint: The application should gracefully handle scenarios with limited or no network connectivity.

Rationale: Users may not always have access to a reliable network, and the application should provide meaningful functionality even in offline or low-connectivity scenarios.

3.2.5 Legal and Regulatory Compliance

Constraint: Ensure that the application complies with relevant legal and regulatory requirements, especially concerning data privacy and user rights.

Rationale: Adherence to legal and regulatory standards is crucial for user trust and avoiding potential legal complications.

4. Use-Case View

The Use-Case View section of the Software Architecture Document (SAD) provides a detailed exploration of the various use cases that drive the functionality and interactions within the Gamify Your Life mobile Android application. This section serves as a comprehensive guide for stakeholders to understand how the application responds to different user interactions and scenarios.

The use cases for the mobile application are:

- Create/Edit/View Tasks.
- Complete tasks and earn credits for successful completion.
- Create/Edit/View Task Lists.
- Group created tasks by task lists.
- Create/Edit/View Habits.
- Earn credits for following habits.
- Create/Edit/View rewards.
- Buy rewards for earned credits.
- Edit tasks settings.
- Edit habits settings.
- Edit rewards settings.
- Search other users by profile name.
- Add users in friends.
- Edit account settings.
- View/Delete friends.
- Share your habits with friends.
- Approve/Decline friend requests.

4.1 Architecturally Significant Use Cases

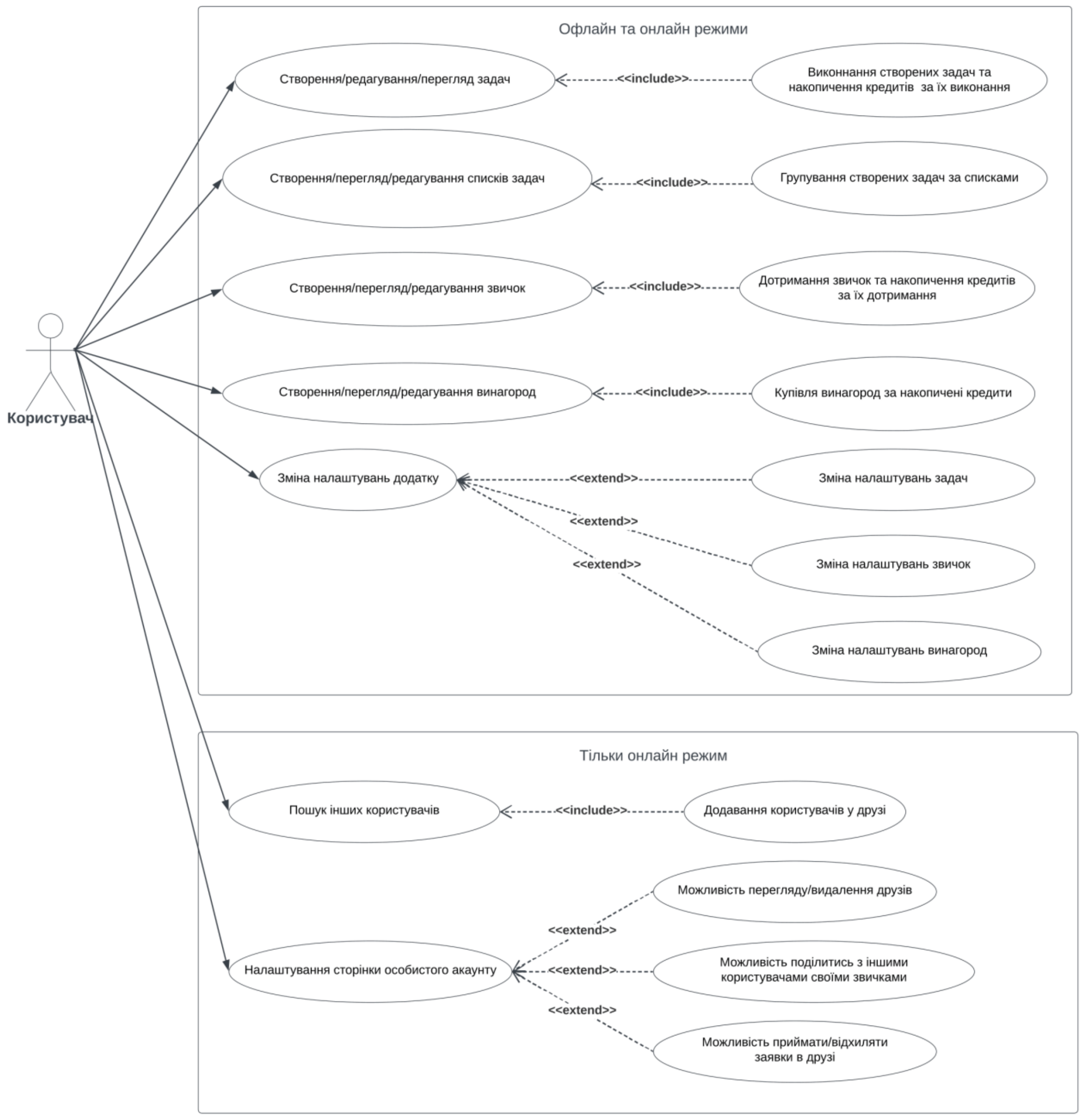
Офлайн та онлайн режими
Створення/редагування/перегляд задач
<<include>>
Виконнання створених задач та накопичення кредитів за їх виконання
Створення/перегляд/редагування списків задач
<<include>>
Групування створених задач за списками
Створення/перегляд/редагування звичок
<<include>>
Дотримання звичок та накопичення кредитів за їх дотримання
Користувач
Створення/перегляд/редагування винагород
<<include>>
Купівля винагород за накопичені кредити
Зміна налаштувань додатку
<<extend>>
Зміна налаштувань задач
<<extend>>
Зміна налаштувань звичок
<<extend>>
Зміна налаштувань винагород
Тільки онлайн режим
Пошук інших користувачів
<<include>>
Додавання користувачів у друзі
Можливість перегляду/видалення друзів
<<extend>>
Налаштування сторінки особистого акаунту
<<extend>>
Можливість поділитись з іншими користувачами своїми звичками
<<extend>>
Можливість приймати/відхиляти заявки в друзі

Figure 4.1: Use-case Diagram

1. **Create/Edit/View Tasks:**
   - **Description:** Users can create, edit, and view individual tasks. Tasks represent specific activities or goals that users aim to accomplish.
2. **Complete Tasks and Earn Credits:**
   - **Description:** Users earn credits upon successfully completing tasks. This feature incentivizes users to stay disciplined and rewards their achievements.
3. **Create/Edit/View Task Lists:**
   - **Description:** Users can organize tasks into lists, providing a structured way to manage and categorize their various responsibilities or objectives.
4. **Group Created Tasks by Task Lists:**
   - **Description:** Users have the ability to group tasks by associating them with specific task lists, facilitating better organization and management of related activities.
5. **Create/Edit/View Habits:**

- **Description:** Users can create, edit, and view habits they want to cultivate. Habits represent repeated actions that contribute to personal development.

6. **Earn Credits for Following Habits:**
   - **Description:** Users accumulate credits by consistently following and completing their defined habits, promoting positive behavior reinforcement.
7. **Create/Edit/View Rewards:**
   - **Description:** Users can set up, modify, and view rewards as a motivational tool. Rewards serve as incentives that users can earn by achieving their goals.
8. **Buy Rewards for Earned Credits:**
   - **Description:** Users have the option to use their earned credits to purchase rewards. This feature enhances the gamification aspect of the application.
9. **Edit Tasks Settings:**
   - **Description:** Users can customize the settings of individual tasks, tailoring them to specific requirements or preferences.
10. **Edit Habits Settings:**
    - **Description:** Users can adjust the settings of their habits, refining parameters to better align with personal preferences or changing circumstances.
11. **Edit Rewards Settings:**
    - **Description:** Users have the flexibility to modify the settings of their rewards, allowing for adjustments in line with evolving preferences or goals.
12. **Search Other Users by Profile Name:**
    - **Description:** Users can search for other Gamify Your Life users by entering profile names, facilitating connections and interactions with friends or acquaintances.
13. **Add Users in Friends:**
    - **Description:** Users can establish connections by sending friend requests to other users, fostering a sense of community within the Gamify Your Life platform.
14. **Edit Account Settings:**
    - **Description:** Users have control over their account settings, allowing them to personalize their experience within the application.
15. **View/Delete Friends:**
    - **Description:** Users can view their list of friends and have the option to delete friends from their network.
16. **Share Your Habits with Friends:**
    - **Description:** Users can share their established habits with friends, encouraging social accountability and support.
17. **Approve/Decline Friend Requests:**
    - **Description:** Users can manage incoming friend requests by either approving or declining them, controlling their network connections.

These use cases collectively form the foundation of the Gamify Your Life application, providing a diverse set of features to support users in enhancing their self-discipline and achieving personal goals.

5. Process View

5.1 Main Business Processes

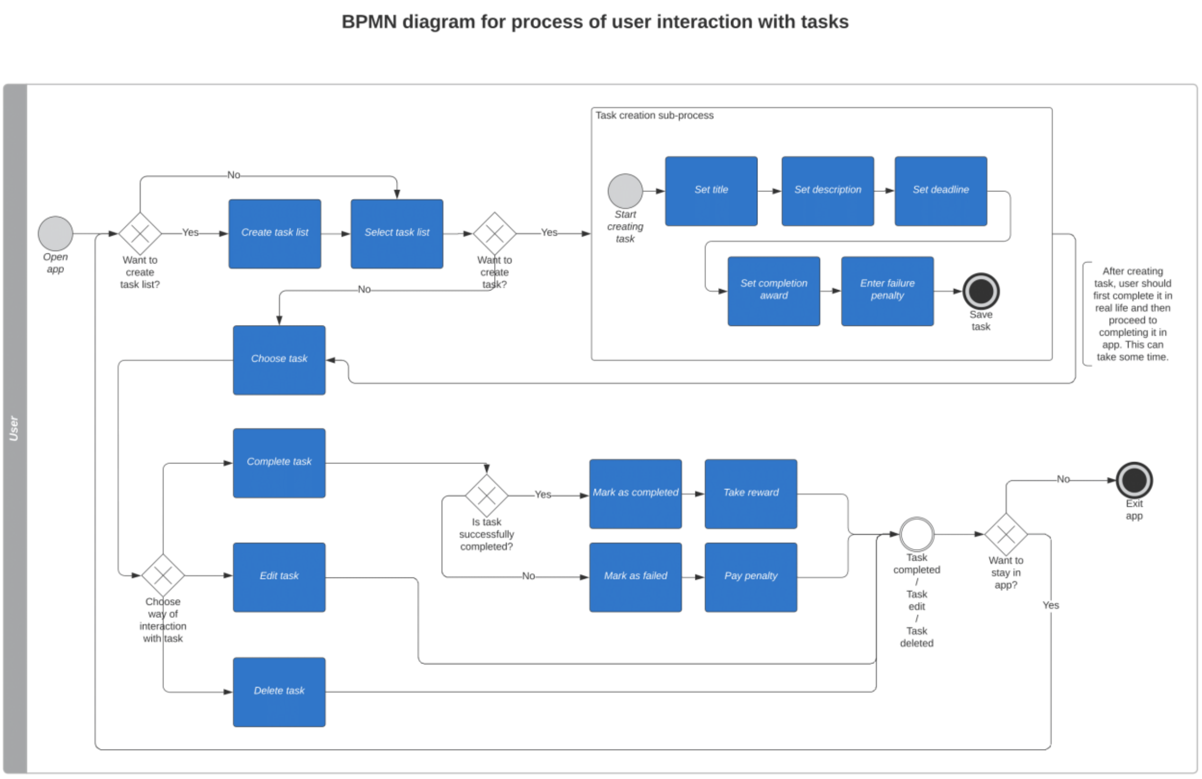
BPMN diagram for process of user interaction with tasks
User
Open app
Want to create task list?
No
Yes
Create task list
Select task list
Want to create task?
Yes
No
Task creation sub-process
Start creating task
Set title
Set description
Set deadline
Set completion award
Enter failure penalty
Save task
After creating task, user should first complete it in real life and then proceed to completing it in app. This can take some time.
Choose task
Complete task
Edit task
Delete task
Choose way of interaction with task
Is task successfully completed?
Yes
No
Mark as completed
Take reward
Mark as failed
Pay penalty
Task completed / Task edit / Task deleted
Want to stay in app?
No
Yes
Exit app

Figure 5.1: BPMN Diagram

Diagram Description:

1. Task List Creation:

- **Activity:** User creates a new task list.
- **Flow:** Start Event → Task List Creation Activity → End Event.

2. Task List Selection:

- **Activity:** User selects a task list from the created lists.
- **Decision:** If the user does not select a task list, loop back to Task List Selection until a valid selection is made.
- **Flow:** Start Event → Task List Selection Activity → Decision (Valid Selection?) → (Yes) End Event, (No) Loop back to Task List Selection.

3. Task Creation:

- **Activity:** User creates a new task within the selected task list.
- **Flow:** Start Event → Task Creation Activity → End Event.

4. Task Interaction:

- **Decision:** User decides whether to edit, mark as completed, mark as failed, or delete the task.
- **Flow:**
    - If the user chooses to edit: Task Interaction Decision (Edit) → Edit Task Activity.

- If the user chooses to mark as completed: Task Interaction Decision (Mark as Completed) → Credits Earned Activity.
- If the user chooses to mark as failed: Task Interaction Decision (Mark as Failed) → Penalty Charged Activity.
- If the user chooses to delete: Task Interaction Decision (Delete) → End Event.

5. Edit Task:

- **Activity:** User edits the details of the selected task.
- **Flow:** Start Event → Edit Task Activity → End Event.

6. Take Reward:

- **Activity:** User take reward for completing the task.
- **Flow:** Start Event → Take Reward Activity → End Event.

7. Penalty Charged:

- **Activity:** User incurs a penalty for marking the task as failed.
- **Flow:** Start Event → Penalty Charged Activity → End Event.

8. End Event:

- **Activity:** Process concludes.

5.2 Message Flow and Program Components Dependencies

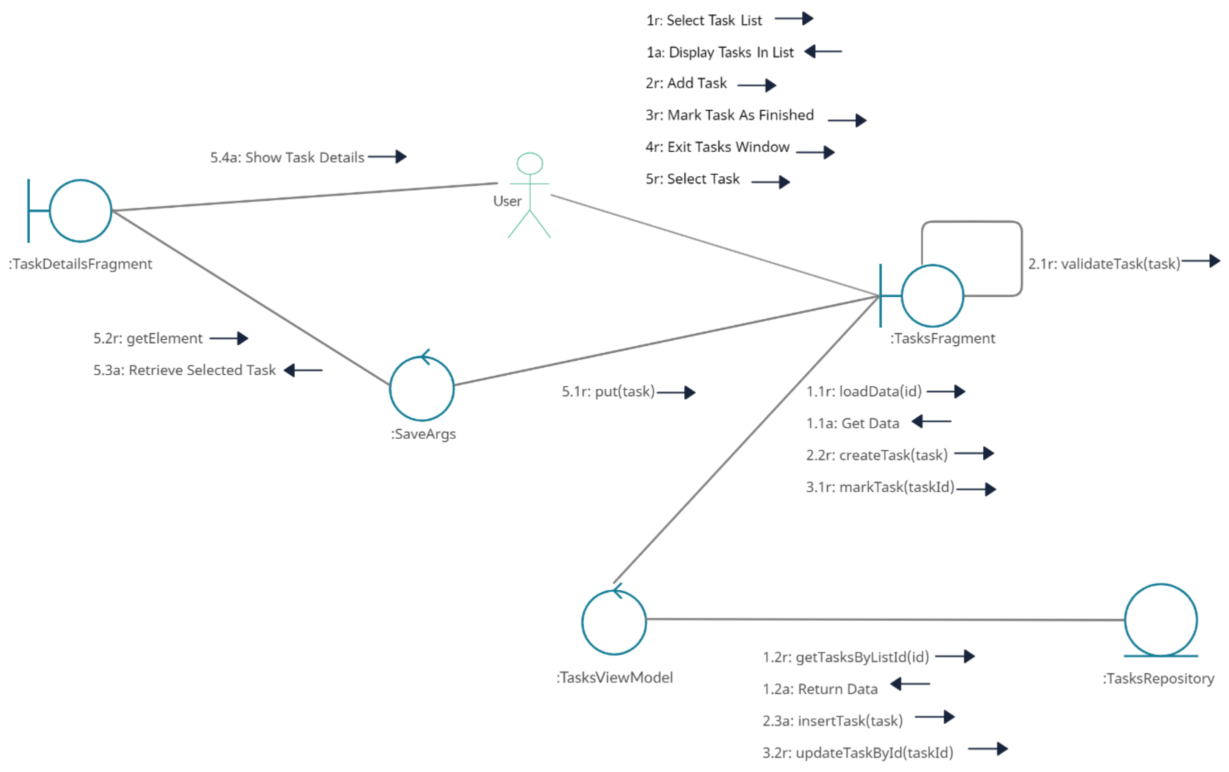

Figure 5.2: Collaboration Diagram

Diagram description:

1. User:

- **Role:** The external actor initiating interactions with the application.
- **Interactions:** Triggers events such as creating, editing, completing, or deleting tasks.

2. TasksFragment:

- **Role:** Represents the Tasks Screen in the MVVM architecture.
- **Interactions:**
    - Receives user input for task-related actions.
    - Communicates with TasksViewModel for business logic operations.
    - Utilizes SaveArgs for data transfer to TaskDetailsFragment.

3. TasksViewModel:

- **Role:** Defines business logic operations for tasks within the MVVM architecture.
- **Interactions:**
    - Receives user-triggered events from TasksFragment.
    - Interacts with TasksRepository for database operations.
    - Communicates with TasksFragment to update the user interface.

4. TasksRepository:

- **Role:** Manages database interactions and serves as the Object-Relational Mapping (ORM) for tasks.
- **Interactions:**
    - Receives requests from TasksViewModel for database operations.
    - Performs CRUD (Create, Read, Update, Delete) operations on Tasks entities.
    - Ensures data integrity and consistency.

5. TaskDetailsFragment:

- **Role:** Represents the Task Details screen in the MVVM architecture.
- **Interactions:**
    - Displays detailed information about a specific task.
    - Communicates with SaveArgs for data transfer to TasksFragment.

6. SaveArgs:

- **Role:** Service responsible for transferring data between TaskDetailsFragment and TasksFragment.
- **Interactions:**
    - Facilitates the exchange of task-related data between TaskDetailsFragment and TasksFragment.
    - Ensures seamless communication during task creation, editing, or other relevant actions.

Collaboration Flow:

1. **User Interaction:**
    - The User triggers events, such as creating, editing, completing, or deleting tasks.
2. **TasksFragment Handling:**
    - TasksFragment receives user input and communicates with TasksViewModel for business logic.
3. **TasksViewModel Operations:**
    - TasksViewModel processes user-triggered events and interacts with TasksRepository for database operations.
4. **TasksRepository Database Operations:**
    - TasksRepository performs necessary database operations, ensuring data integrity.

5. **TasksViewModel Update:**
    - TasksViewModel updates TasksFragment with the latest information to reflect changes in the user interface.
6. **TaskDetailsFragment Display:**
    - If the user opts to view detailed information about a task, TaskDetailsFragment is displayed.
7. **Data Transfer via SaveArgs:**
    - SaveArgs facilitates seamless data transfer between TaskDetailsFragment and TasksFragment, ensuring consistent information.

5.3 Class relations

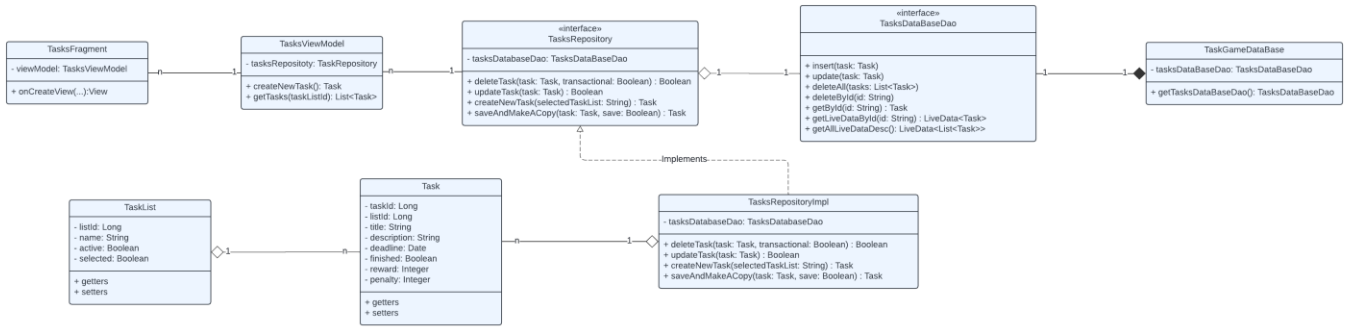
TasksFragment
- viewModel: TasksViewModel
+ onCreateView(...):View
n
1
TasksViewModel
- tasksRepositoty: TaskRepository
+ createNewTask(): Task
+ getTasks(taskListId): List<Task>
n
1
«interface»
TasksRepository
- tasksDatabaseDao: TasksDataBaseDao
+ deleteTask(task: Task, transactional: Boolean) : Boolean
+ updateTask(task: Task) : Boolean
+ createNewTask(selectedTaskList: String) : Task
+ saveAndMakeACopy(task: Task, save: Boolean) : Task
1
1
«interface»
TasksDataBaseDao
+ insert(task: Task)
+ update(task: Task)
+ deleteAll(tasks: List<Task>)
+ deleteById(id: String)
+ getById(id: String) : Task
+ getLiveDataById(id: String) : LiveData<Task>
+ getAllLiveDataDesc(): LiveData<List<Task>>
1
1
TaskGameDataBase
- tasksDataBaseDao: TasksDataBaseDao
+ getTasksDataBaseDao(): TasksDataBaseDao
Implements
TaskList
- listId: Long
- name: String
- active: Boolean
- selected: Boolean
+ getters
+ setters
1
n
Task
- taskId: Long
- listId: Long
- title: String
- description: String
- deadline: Date
- finished: Boolean
- reward: Integer
- penalty: Integer
+ getters
+ setters
n
1
TasksRepositoryImpl
- tasksDatabaseDao: TasksDatabaseDao
+ deleteTask(task: Task, transactional: Boolean) : Boolean
+ updateTask(task: Task) : Boolean
+ createNewTask(selectedTaskList: String) : Task
+ saveAndMakeACopy(task: Task, save: Boolean) : Task

Figure 5.3: UML Class Diagram

Diagram description:

1. **TasksFragment Class:**
    - **Methods:**
        - **onCreateView()**: Overrides the onCreateView method to initialize the fragment's user interface.
2. **TasksViewModel Class:**
    - **Methods:**
        - **createNewTask()**: Initiates the process of creating a new task.
        - **getTasks(taskListId)**: Retrieves a collection of tasks associated with a specific task list.
3. **Task Class (Simple Entity):**
    - Represents a simple entity class that encapsulates information about an individual task.
4. **TaskList Class (Entity):**
    - Represents a collection of tasks. Each TaskList contains multiple Task entities.
5. **TaskRepository Interface:**
    - **Methods:**
        - **deleteTask(taskId)**: Deletes a task based on its unique identifier.
        - **updateTask(taskId)**: Updates the details of a task.
        - **createNewTask(task, taskListId)**: Creates a new task and associates it with a specific task list.
        - **saveAndMakeACopy(task)**: Saves the task and creates a duplicate.
6. **TaskRepositoryImpl Class:**
    - **Implementation of TaskRepository Interface:**
        - Implements the methods specified in the TaskRepository interface.
7. **TaskDataBaseDao Interface:**
    - **Methods:**
        - **insert(task)**: Inserts a new task into the database.
        - **update(task)**: Updates the details of an existing task.
        - **deleteAll()**: Deletes all tasks from the database.
        - **deleteById(taskId)**: Deletes a task based on its unique identifier.
        - **getById(taskId)**: Retrieves a task based on its unique identifier.

8. **TaskGameDataBase Class:**
    - **Methods:**
        - **getTaskDataBaseDao()**: Retrieves the Data Access Object (DAO) for task-related database operations.

Class Relationships:

- **TasksFragment and TasksViewModel:**
    - TasksFragment interacts with TasksViewModel to handle user interface updates and user input processing.
- **TasksViewModel and TaskRepository Interface:**
    - TasksViewModel relies on the TaskRepository interface for executing various database-related operations.
- **TaskRepository Interface and TaskRepositoryImpl Class:**
    - TaskRepositoryImpl provides the concrete implementation of the TaskRepository interface, defining the actual logic for database operations.
- **TaskRepositoryImpl Class and TaskDataBaseDao Interface:**
    - TaskRepositoryImpl interacts with TaskDataBaseDao to perform database operations.
- **TaskDataBaseDao Interface and TaskGameDataBase Class:**
    - TaskGameDataBase provides the DAO for accessing and manipulating task-related data within the database.

6. Deployment View

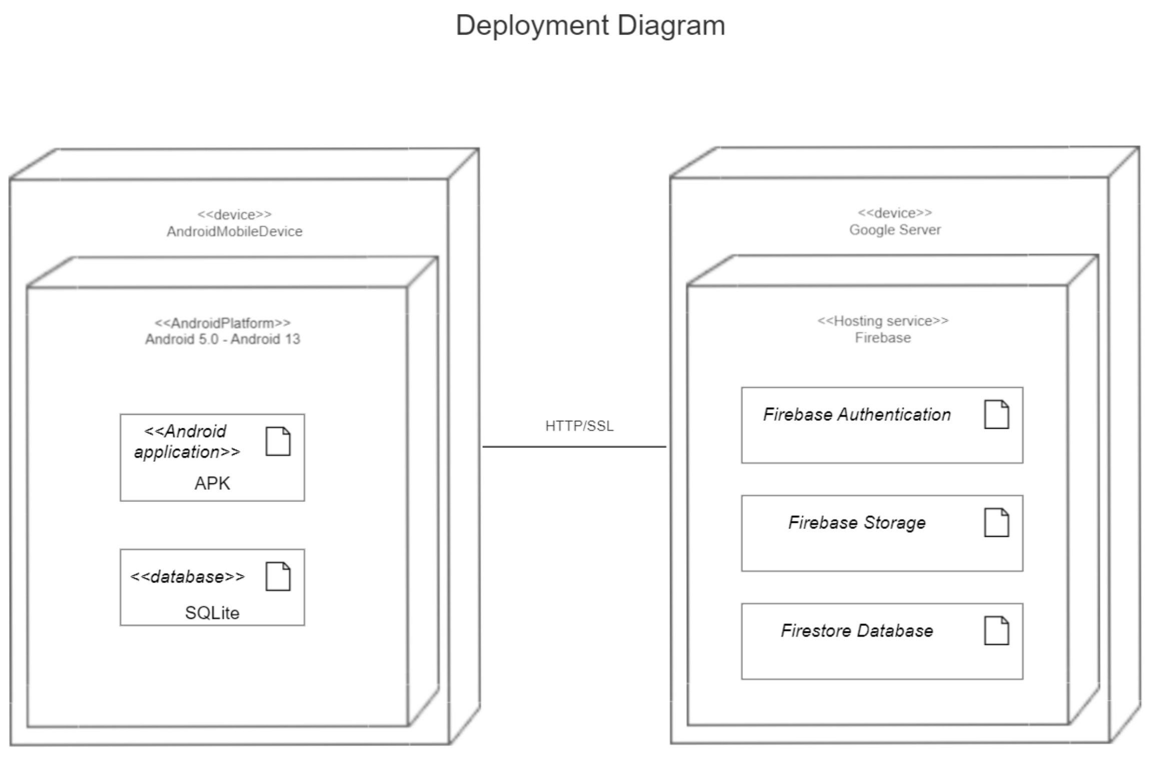

Figure 6: UML Deployment Diagram

Diagram description:

1. **Android Mobile Device:**
    - Represents the physical or virtual device where the Gamify Your Life Android application runs.

    - Hosts the Android Platform, which includes the Android Application APK and SQLite database.
2. **Android Platform: Android 5.0 - Android 13:**
    - Represents the Android operating system running on the Android Mobile Device.
    - Hosts the Gamify Your Life Android application in the form of an APK (Android Package) file.
    - Manages the local SQLite database for storing app-related data.

3. **Google Service - Device:**
    - Represents the physical or virtual server device hosting the backend services for Gamify Your Life.
    - Runs the Firebase hosting service and serves as the connection point for frontend-backend communication.
4. **Firebase Hosting Service:**
    - Provides hosting for the Gamify Your Life backend services.
    - Includes several Firebase services:
        - **Firebase Authentication:** Manages user authentication and authorization.
        - **Firebase Storage:** Handles storage of user-related data, possibly media files or user profile images.
        - **Firestore Database:** Serves as the backend database for storing and retrieving application data.

Communication:

5. **HTTP/SSL:**
    - Represents the secure communication protocol used for data exchange between the frontend (Android Mobile Device) and the backend (Google Service with Firebase hosting services).
    - Ensures the confidentiality and integrity of data during transmission.

Deployment Relationships:

- The Android Mobile Device interacts with the Android Platform, which includes the Gamify Your Life application and SQLite database.
- The Android Platform communicates with the Google Service - Device, hosting the Firebase services.
- The Firebase Hosting Service, which includes Firebase Authentication, Firebase Storage, and Firestore Database, collaborates to handle various backend functionalities.
- Communication between the frontend and backend is established using HTTP/SSL to ensure secure data transmission.

7. Size and Performance

7.1 Size Considerations

The Gamify Your Life mobile application is designed with a focus on optimizing size to enhance user accessibility and download efficiency. The APK file size is carefully managed to ensure a quick and seamless installation process for users, particularly those with limited storage capacity or slower network connections. The use of efficient coding practices, resource optimization, and judicious inclusion of features contribute to a compact application footprint.

The size of media assets, such as images and icons, is also optimized to strike a balance between visual appeal and overall application size. This approach aims to deliver a user-friendly experience without compromising on the application's performance and responsiveness.

7.2 Performance Considerations
Performance is a critical aspect of the Gamify Your Life application to ensure a smooth and responsive user experience. Several performance considerations are integrated into the application's design and development:

7.2.1 Responsiveness:

- User interactions, such as task creation, editing, and completion, are designed to be responsive, providing immediate feedback to enhance the perceived speed of the application.

7.2.2 Network Efficiency:

- The application optimizes network usage to minimize data transfer, especially in scenarios with limited connectivity. Asynchronous operations are employed to prevent user interface freezing during network requests.

7.2.3 Database Optimization:

- The SQLite database is structured and queried efficiently to reduce latency in data retrieval and manipulation. Indexing and caching mechanisms are implemented judiciously to enhance database performance.

7.2.4 Code Efficiency:

- The source code is organized and written with a focus on efficiency. Algorithms and data structures are chosen to optimize processing speed, ensuring that the application operates smoothly even on devices with varying hardware capabilities.

7.2.5 Memory Management:

- Strict memory management practices are employed to prevent memory leaks and excessive resource consumption. This ensures a stable application that performs well across a range of devices.

7.2.6 Battery Optimization:

- The application is designed to minimize battery consumption, with features like background services optimized to run efficiently without unduly impacting the device's power resources.

7.3 Performance Testing
The Gamify Your Life application undergoes rigorous performance testing to identify and address potential bottlenecks, memory leaks, and responsiveness issues. Testing scenarios include varying network conditions, different device specifications, and simulated usage patterns to ensure optimal performance across a diverse user base.

By addressing both size and performance considerations, the Gamify Your Life application aims to deliver a lightweight yet powerful tool for users to enhance self-discipline, with a seamless and efficient user experience on a broad spectrum of Android devices.

8. Quality

The Gamify Your Life mobile application is committed to delivering a high-quality user experience by adhering to industry-standard best practices and principles. The development process incorporates rigorous testing, code reviews, and continuous improvement to ensure the following key aspects of quality:

8.1 Reliability and Stability:
The application undergoes thorough testing for stability and reliability, aiming to minimize crashes, errors, and unexpected behavior. Regular updates and bug fixes are scheduled to address issues promptly, ensuring a dependable user experience.

8.2 Usability:
The user interface is designed with a focus on usability, providing an intuitive and efficient experience for users. Usability testing is conducted to gather feedback and make iterative improvements to the interface, ensuring ease of navigation and task completion.

8.3 Security:
Security is a top priority, and the application employs industry-standard practices to safeguard user data and privacy. This includes secure communication protocols, data encryption, and adherence to best practices for user authentication and authorization.

8.4 Maintainability:
The codebase is structured and documented to facilitate ease of maintenance. Regular code reviews and version control practices are implemented to ensure that the application remains maintainable and adaptable to future enhancements or updates.

8.5 Performance:
The application is optimized for performance, aiming to deliver a responsive and efficient user experience. Performance testing is conducted under various conditions to identify and address potential bottlenecks, ensuring optimal performance across a diverse range of devices.

8.6 Scalability:
The architecture is designed to be scalable, allowing the application to accommodate a growing user base and additional features seamlessly. Scalability testing is conducted to ensure that the application performs effectively as user demand increases.

8.7 Compliance:
The Gamify Your Life application adheres to relevant legal and regulatory standards, particularly concerning data privacy and user rights. Compliance measures are continually reviewed and updated to align with evolving regulations.

By prioritizing these quality considerations, Gamify Your Life aims to provide users with a reliable, secure, and enjoyable experience while promoting self-discipline and personal development.