TECHNICAL DOCUMENTATION

# Table of Contents

1. INTRODUCTION

Project Overview

The TrellTech Project Management App is designed to streamline and enhance the management of software development projects by leveraging the Trello API. It facilitates task tracking, collaboration, and project planning in a user-friendly interface. The application aims to simplify the complexities involved in project management by offering a suite of tools that enable users to create, update, delete, and display workspaces, boards, lists, and cards. Additionally, it supports assigning persons to a card, ensuring a clear distribution of responsibilities among team members. This app is intended to serve as a bridge between the conceptual planning stages of project management and the practical execution of tasks, thereby increasing productivity and project visibility.

Target Platform

FLUTTER

Choosing Flutter as the framework for developing the TRELLTECH Project Management App was a strategic decision driven by several compelling reasons that align perfectly with the project's goals and constraints. Flutter stands out as an innovative framework for building high-quality, natively compiled applications for mobile, web, and desktop from a single codebase. Here are the reasons why Flutter was the ideal choice for this project:

Cross-Platform Development

Flutter's ability to deliver a truly native experience across multiple platforms from a single codebase is unparalleled. This cross-platform capability ensures that TRELLTECH can be deployed on both iOS and Android without significant changes to the underlying code. This efficiency in development significantly reduces time to market and development costs while maintaining a consistent user experience across platforms.

Performance

Flutter compiles to native code, which is crucial for achieving optimal performance. Unlike other cross-platform frameworks that rely on bridge technologies or web views, Flutter's direct compilation to native code ensures that TRELLTECH benefits from the full performance capabilities of the underlying platform. This results in smooth animations, a responsive UI, and an overall fluid experience that matches or exceeds that of native applications.

Rich Set of Widgets and Customizable UI

Flutter provides a comprehensive set of highly customizable widgets that adhere to Material Design and Cupertino (iOS-flavor) guidelines, enabling the creation of beautiful and highly interactive user interfaces. For TRELLTECH, this means the ability to craft a polished and engaging UI that enhances the project management experience for users. Furthermore, the extensive widget library simplifies the development process, as many common UI components are readily available and easily customizable to fit the app's needs.

Strong Community Support and Ecosystem

Flutter's growing community and ecosystem offer a wealth of resources, including packages, tools, and plugins that extend its capabilities. For TRELLTECH, leveraging these resources means access to a wide range of functionalities that can be easily integrated into the app, such as connectivity with the Trello API, authentication services, and state management solutions. This ecosystem not only enriches the app's feature set but also reduces the need to develop complex functionalities from scratch.

In summary, Flutter's blend of performance, developer productivity, visual appeal, and cross-platform reach made it the optimal choice for developing the TRELLTECH Project Management App. Its ability to deliver a high-quality, native-like user experience across multiple platforms, coupled with the efficiency of development and the support of a vibrant ecosystem, aligns perfectly with the project's objectives.

2. ARCHITECTURE OVERVIEW

ARCHITECTURAL PATTERN

For the TRELLTECH Project Management App, we've adopted the Model-View-Controller (MVC) architecture pattern as the foundation of our Flutter application's design. This decision is underpinned by the MVC pattern's robust separation of concerns, facilitating a modular and organized approach to application development that aligns seamlessly with Flutter's widget-centric framework.

Model

The Model component represents our application's dynamic data structure, isolated from user interfaces. It directly manages the data, logic, and rules of the application. In the context of the TRELLTECH app, Models are responsible for encapsulating the attributes of workspaces, boards, lists, and cards, alongside their interactions with the Trello API. This includes data retrieval, content creation, updates, and deletions—actions that are fundamental to the app's core functionality.

View

The View component is tasked with presenting the model data to users. It's a visual representation of the models, rendering the user interface based on the data received from the Model. Flutter's declarative UI philosophy amplifies the effectiveness of the View component, enabling dynamic and responsive designs that reflect changes in the model to the user in real time. Our application leverages custom widgets and pre-built Flutter components to craft a user interface that's not only intuitive but also aligns with modern design principles, enhancing user experience and engagement.

Controller

The Controller acts as an intermediary between the Model and View, receiving user input and mediating the data flow between them. In our application, Controllers handle user interactions, interpret them, and then manipulate the data model accordingly. Subsequently, the model updates the view, ensuring the user interface is in sync with the underlying data. This encapsulation of business logic within the Controller allows for a clear distinction between the application's operational logic and its presentation layer.

INTEGRATION WITH THE TRELLO API

The TRELLTECH app extensively utilizes the Trello API to offer a comprehensive project management tool. MVC architecture significantly simplifies the integration process, enabling efficient handling of API requests and responses within the Model component. This separation ensures that our application logic concerning Trello's services is well-organized and easily maintainable, facilitating future enhancements and updates without impacting the UI or the application's core functionality.

Advantages for TRELLTECH

Choosing the MVC architecture for the TRELLTECH Project Management App offers several advantages:

Modularity: By separating concerns, we enhance the app's modularity, making it easier to update, maintain, and scale. This structure allows individual team members to work on the Model, View, or Controller independently, boosting development efficiency.

Reusability: MVC promotes the reuse of components. Our custom Flutter widgets, designed under the View component, can be easily reused across different parts of the application, ensuring a consistent look and feel while reducing development time.

Testability: With clear separation between business logic, data models, and presentation, testing becomes more straightforward. Each component can be independently unit tested, ensuring robustness and reliability.

Flexibility in Design and Development: MVC architecture allows designers to focus on the View component without needing to understand the underlying code structure fully. Similarly, developers can concentrate on the business logic and data management, knowing that changes to the logic or data models won't directly impact the UI design.

In conclusion, the MVC architecture offers a structured approach to developing the TRELLTECH Project Management App with Flutter. It aligns with our goals of creating a scalable, maintainable, and user-friendly application that leverages the Trello API to deliver a seamless project management experience.

DIRECTORY STRUCTURE

The directory structure of the TrellTech Project Management App is organized to facilitate easy navigation and scalability of the project. Below is an overview of the key directories and their intended purposes:

/android and /ios: These directories contain platform-specific code for Android and iOS, respectively. They are automatically generated by Flutter and can be modified to customize native code, plugins, and platform-specific settings.

/lib: This is the main directory where the Dart code resides. It includes the application's logic, screens, widgets, and services. The structure within the /lib directory is further organized as follows:

/models: Contains the data models used by the application, representing the structure of objects passed between the frontend and backend or used within the app.

/views or /screens: Holds the UI code for each screen in the application. Screens are the main components that represent different pages or views in the app.

/widgets: Contains reusable UI components that are used across multiple screens/views. This promotes code reuse and consistency throughout the app.

/services: Includes classes and functions that handle business logic, such as API calls, data manipulation, and interaction with external services or databases.

/utils or /helpers: Contains utility functions and constants that are used across the application. This can include theme data, form validators, or conversion utilities.

/controllers or /providers: (Depending on the state management approach used) This directory houses the logic for state management, facilitating communication between the app's UI and the business logic.

/assets: Stores static assets used by the application, such as images, icons, and fonts. It's divided into subdirectories for better organization (e.g., /images, /icons).

/test: Contains all unit, widget, and integration tests for the application. It's structured mirroring the /lib directory to simplify locating tests for specific components or functionalities.

pubspec.yaml: A crucial file at the root of the Flutter project that specifies the Flutter SDK version, dependencies (external packages the app uses), and project metadata (such as the app name, description, and version). It also configures assets and fonts that the application uses.

This structured approach ensures that developers can easily locate and manage the application's codebase as it grows in complexity. By adhering to this structure, the project remains maintainable and scalable, accommodating new features and adjustments with minimal disruption.

3. ENVIRONMENT SETUP

To contribute to or build the TRELLTECH Project Management App, ensure your development environment meets the following requirements:

Flutter & Dart Version

Flutter Version: 3.19.1 (Stable channel)

Dart Version: 3.3.0

These versions were used during the development of TRELLTECH, ensuring compatibility and stability of the features implemented.

Dependencies

LIBRARIES AND PACKAGES

Based on my pubspec.yaml content for this Flutter project named "trello_wish," we have used several libraries/packages, each serving different purposes in our application. Here's a breakdown of what each one is typically used for:

english_words (v4.0.0): A Flutter package that provides English words for generating text, often used in examples or for generating placeholder content.

provider (v6.1.2): A state management solution that is widely used in Flutter applications to efficiently manage the app's state and facilitate communication between different parts of the app.

flutter_web_auth (v0.5.0): A package that allows you to perform web-based authentication in a Flutter app. This is useful for integrating OAuth and other web-based authentication methods.

http (v1.2.1): A package for making HTTP requests in Flutter. It's used for networking, allowing your app to communicate with REST APIs or any web services.

flutter_secure_storage (v9.0.0): A package for securely storing data in keychain and keystore. It's typically used to store sensitive information securely, such as tokens, user credentials, or personal data.

flutter_test: Comes with the Flutter SDK and provides a way to write test cases for your Flutter applications. It's part of the dev_dependencies, indicating it's only used in the development environment for testing.

flutter_lints (v3.0.1): A package of Flutter-specific lint rules that encourage good coding practices. It's also a dev_dependency, meaning it's used during development to analyze the code for potential errors, stylistic issues, and code quality improvements.

Flutter SDK: The core SDK for developing Flutter applications. It provides the framework, widgets, and tools needed to build apps for iOS, Android, web, and desktop from a single codebase.

Dart SDK: Specified by the environment field, which requires a Dart version of at least 3.1.1. Dart is the programming language used to write Flutter apps.

Build Instructions

Follow these steps to set up your development environment and build the TRELLTECH app:

Install Flutter and Dart: If you haven't already, download and install Flutter and Dart on your system. Follow the official Flutter installation guide.

Clone the Repository:

```
git clone https://github.com/yourusername/trelltech.git
```

Replace yourusername with your GitHub username if you have forked the project or the original repository URL.

Navigate to the Project Directory:

sh

```
cd trelltech
```

Install Dependencies:

Run the following command to fetch and install all the required dependencies as defined in pubspec.yaml:

sh

```
flutter pub get
```

Run the Application:

Connect your device or emulator and execute the following command to run the app:

sh

```
flutter run
```

This setup guide ensures that developers can clone, build, and run the TRELLTECH app with ease, facilitating a smooth development process.

4.TRELLO API INTEGRATION IN TRELLTECH

Authentication

To ensure secure access to the Trello API, we implemented token-based authentication. This method involves using a combination of an API key and a token. The API key identifies the application making the request, while the token grants specific permissions to access and modify resources within Trello on behalf of the user. Here's an example of how we've structured the authentication:

dart

Copy code

```
static const String apiKey = '68e9b7f0a24622a0ecb4a3177b825720';

static const String apiToken = 'ATTAeddaa663a4434ac939f29595e0cc78eaf68af7d50cbd7f467897e0125caa756c9EA2F2EC';
```

This approach ensures that all interactions with the Trello API are secure and authorized, aligning with best practices for API integration and data protection.

Making Requests

Our interactions with the Trello API are encapsulated within a dedicated class, TrelloAPI, which handles the construction and execution of HTTP requests to Trello's endpoints. Here's a glimpse into how we create a new board on Trello:

dart

Copy code

```
static Future<void> createTrelloBoard(String boardName) async {

  final Uri url = Uri.parse('https://api.trello.com/1/boards/?name=$boardName&key=$apiKey&token=$apiToken');

  try {

    final response = await http.post(url);

    if (response.statusCode == 200) {
```

```
    var data = json.decode(response.body);
    print("Board created successfully: ${data['name']} with ID ${data['id']}");
   } else {
    print("Error creating board: ${response.body}");
   }
  } catch (e) {
   print("Exception while creating board: $e");
  }
 }
```

This method demonstrates our procedure for creating a board by sending a POST request to the Trello API, including handling responses and errors. The use of asynchronous programming ensures that our app remains responsive, enhancing the user experience.

Endpoint Usage

The TRELLTECH app leverages various Trello API endpoints to provide a comprehensive project management tool. Key functionalities include:

Creating, updating, deleting, and displaying workspaces: Achieved by interacting with Trello's organization and board endpoints.

Managing boards: Utilizing board endpoints to create templates (e.g., Kanban), and perform CRUD operations.

Lists and cards management: Through list and card endpoints, enabling users to organize their tasks effectively.

Assigning persons to a card: This involves the members endpoint, allowing for collaboration among team members.

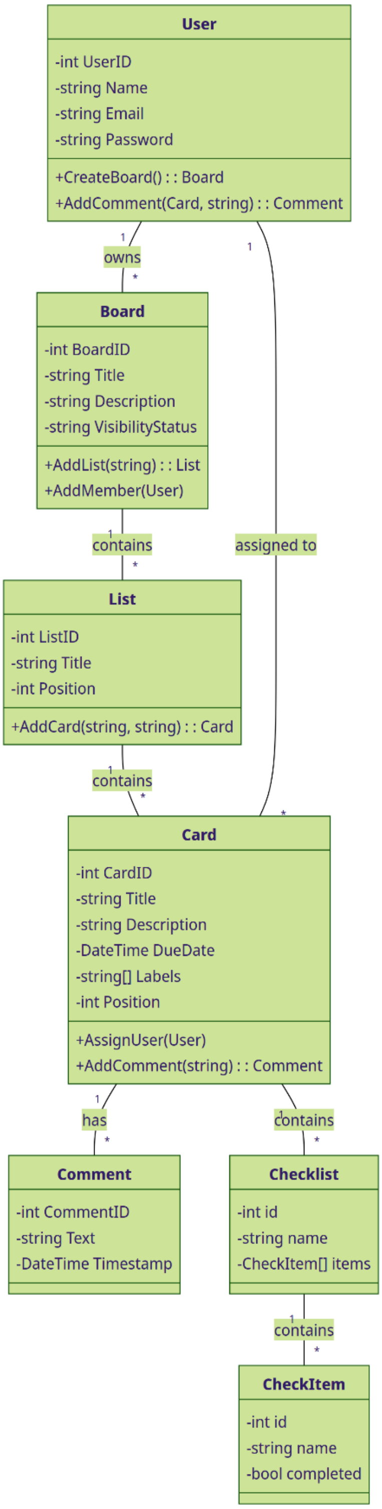
Conclusion

The integration with the Trello API is a cornerstone of the TRELLTECH app, enabling us to offer a rich set of project management features. By adhering to best practices for API interaction, authentication, and error handling, we ensure that TRELLTECH provides a seamless and secure experience for managing projects efficiently and effectively. This methodology underscores our commitment to leveraging existing technologies to enhance productivity and collaboration within teams.

6.Diagrams

Class Diagrams

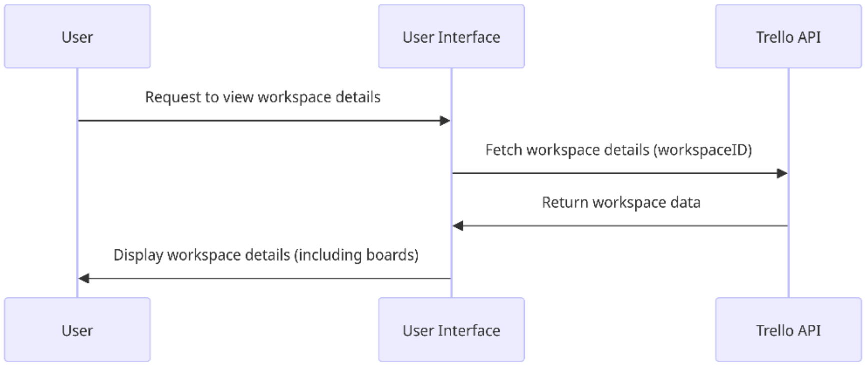
The class diagrams section provides an overview of the TRELLTECH app’s structure, showcasing the relationships between different classes. These visual representations help developers quickly understand the app's design and how its components interact.



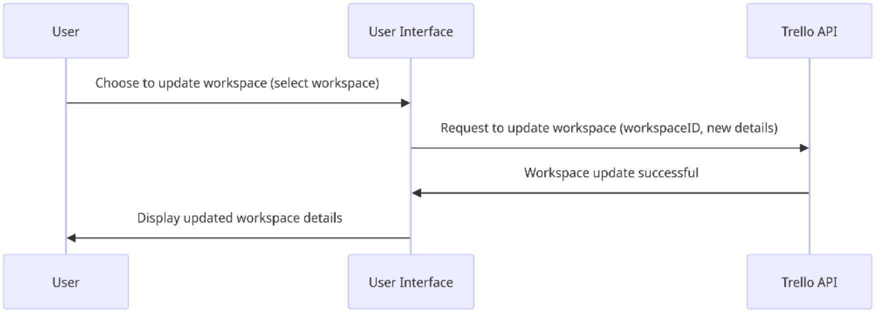
Sequence Diagrams

Sequence diagrams in this section detail the flow of operations within the TRELLTECH app. They illustrate the interactions between objects across time, clarifying the process behind each feature and enhancing the understanding of the app's functionality.

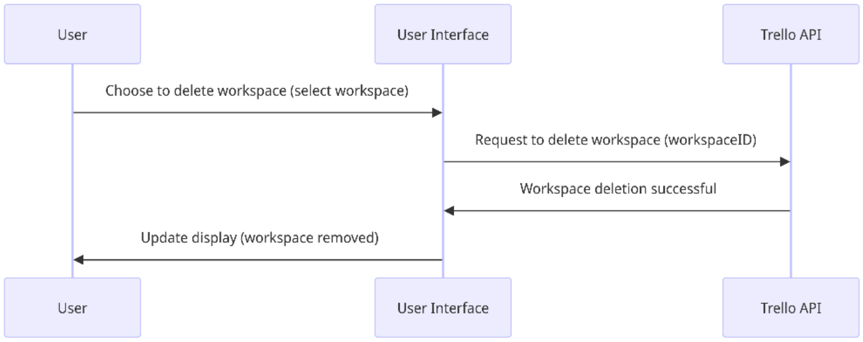
Displaying Workspace


User
User Interface
Trello API
Request to view workspace details
Fetch workspace details (workspaceID)
Return workspace data
Display workspace details (including boards)
User
User Interface
Trello API

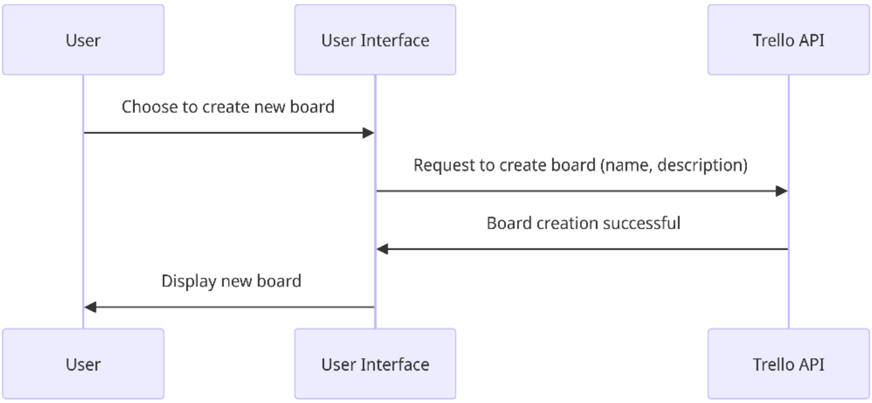
Updating a Workspace


User
User Interface
Trello API
Choose to update workspace (select workspace)
Request to update workspace (workspaceID, new details)
Workspace update successful
Display updated workspace details
User
User Interface
Trello API

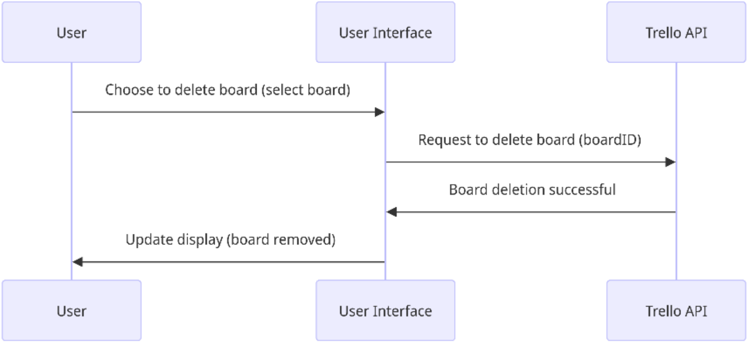
Deleting a Workspace


User
User Interface
Trello API
Choose to delete workspace (select workspace)
Request to delete workspace (workspaceID)
Workspace deletion successful
Update display (workspace removed)
User
User Interface
Trello API

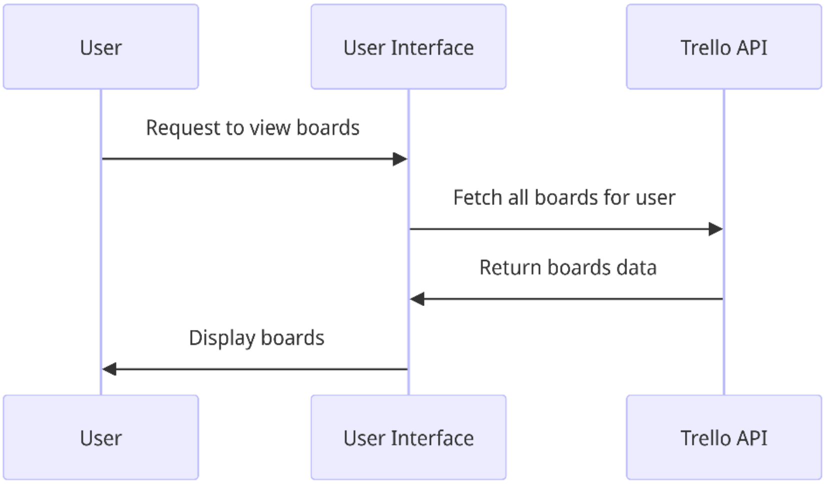
Creating a Board


User
User Interface
Trello API
Choose to create new board
Request to create board (name, description)
Board creation successful
Display new board
User
User Interface
Trello API

Deleting a Board

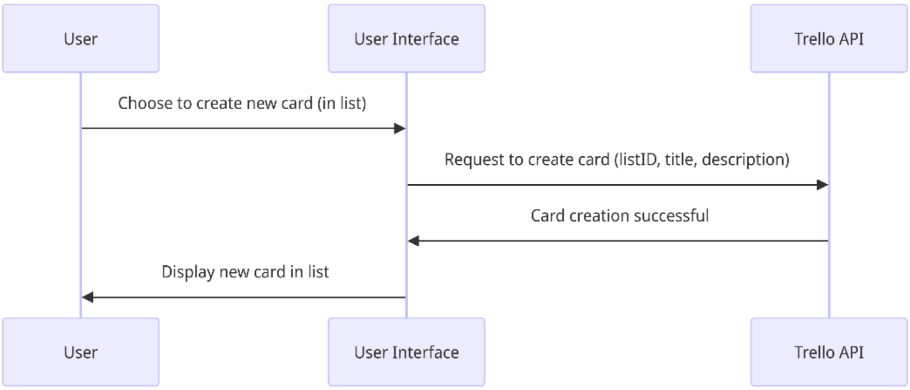

User
User Interface
Trello API
Choose to delete board (select board)
Request to delete board (boardID)
Board deletion successful
Update display (board removed)
User
User Interface
Trello API

Displaying Boards

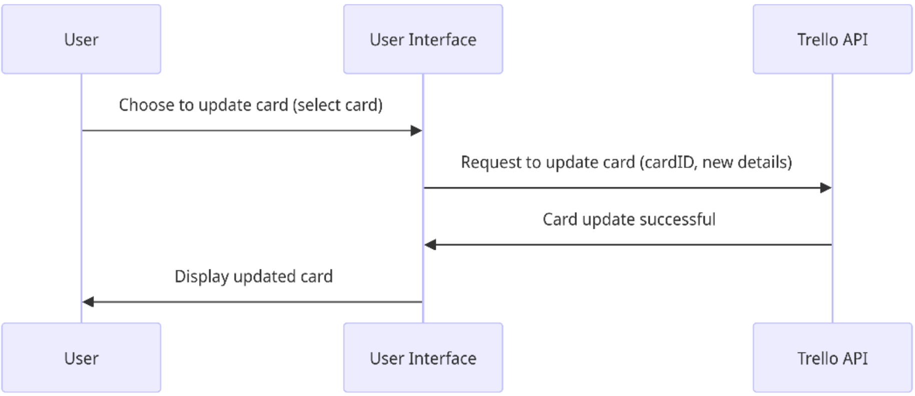


CARD

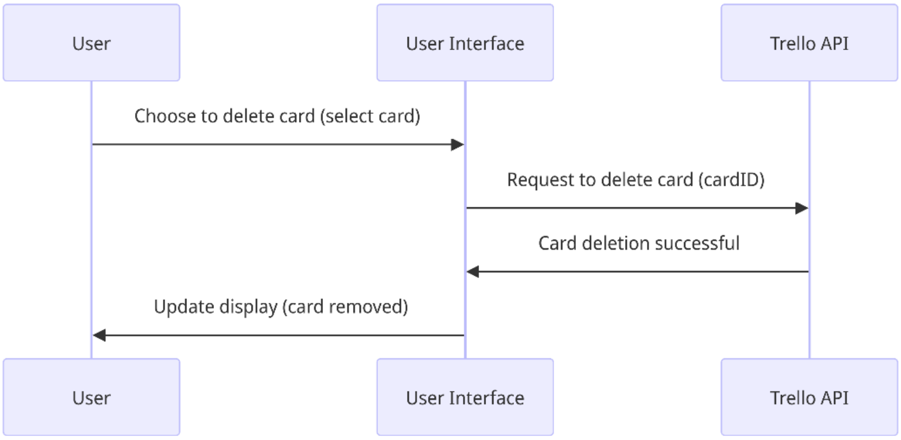
Creating a Card



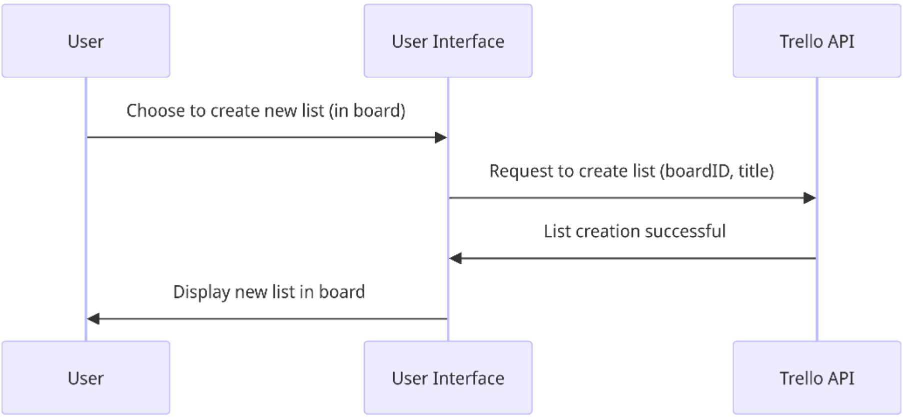
Updating a Card



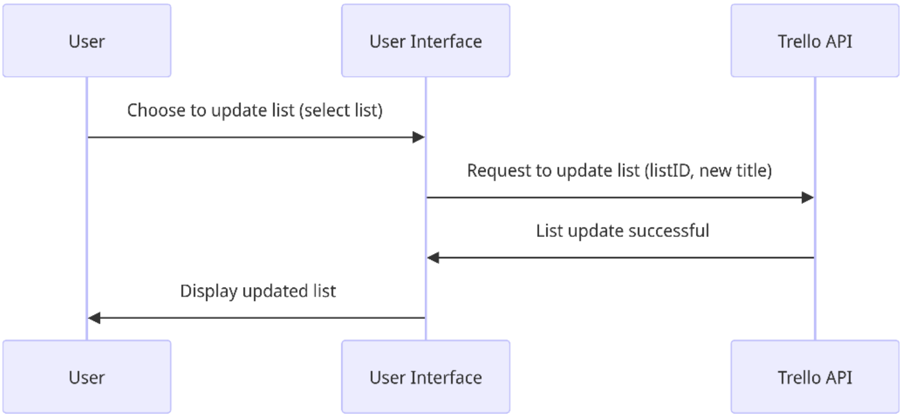
Deleting a Card


User
User Interface
Trello API
Choose to delete card (select card)
Request to delete card (cardID)
Card deletion successful
Update display (card removed)
User
User Interface
Trello API

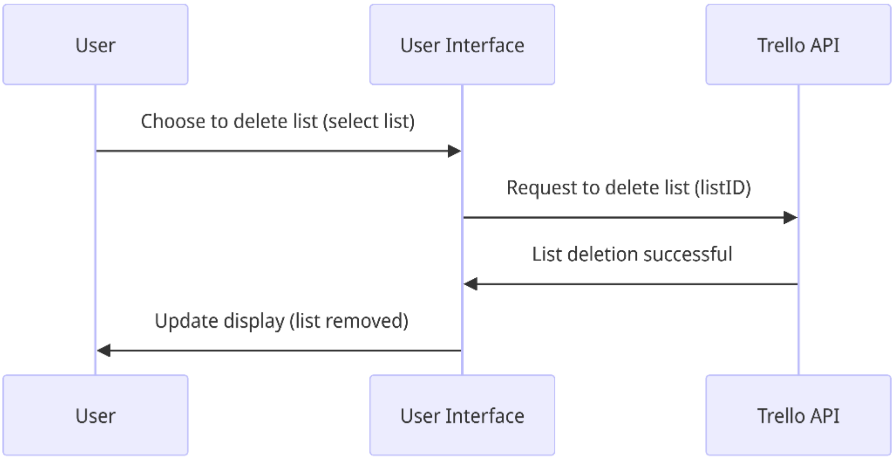
Creating a List


User
User Interface
Trello API
Choose to create new list (in board)
Request to create list (boardID, title)
List creation successful
Display new list in board
User
User Interface
Trello API

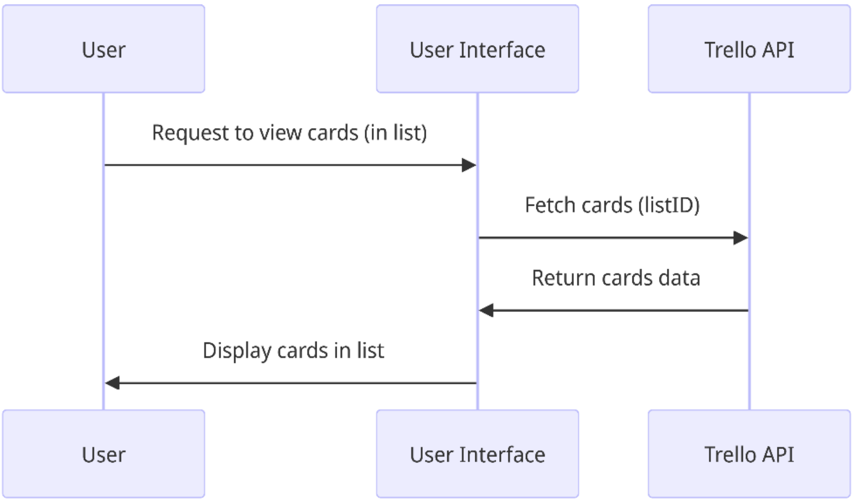
Updating a List



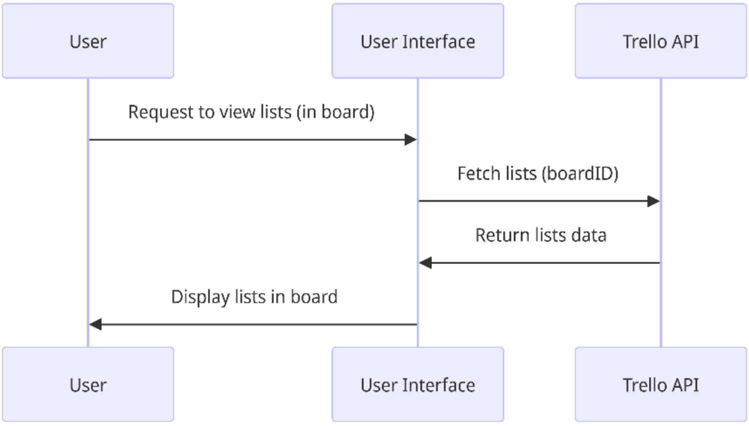
Deleting a List


User
User Interface
Trello API
Choose to delete list (select list)
Request to delete list (listID)
List deletion successful
Update display (list removed)
User
User Interface
Trello API

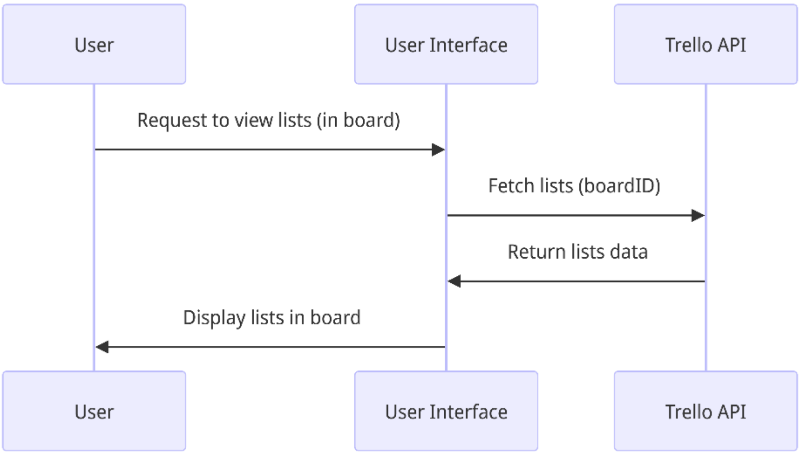
Displaying Cards in a List



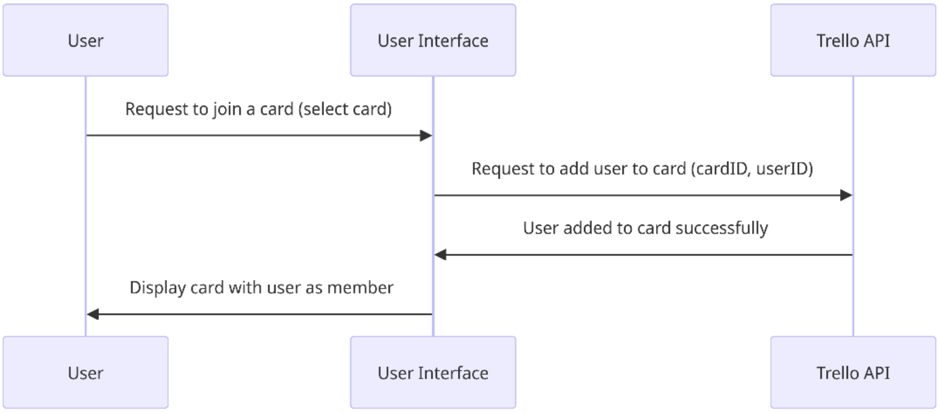
Displaying Lists in a Board



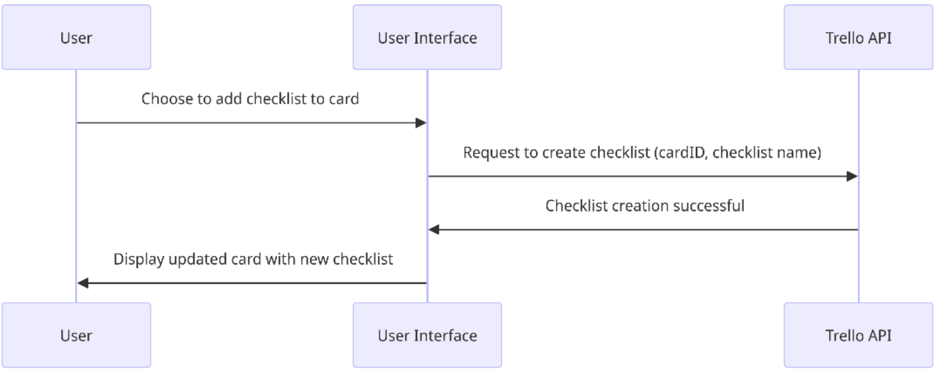
Adding a Comment to a Card



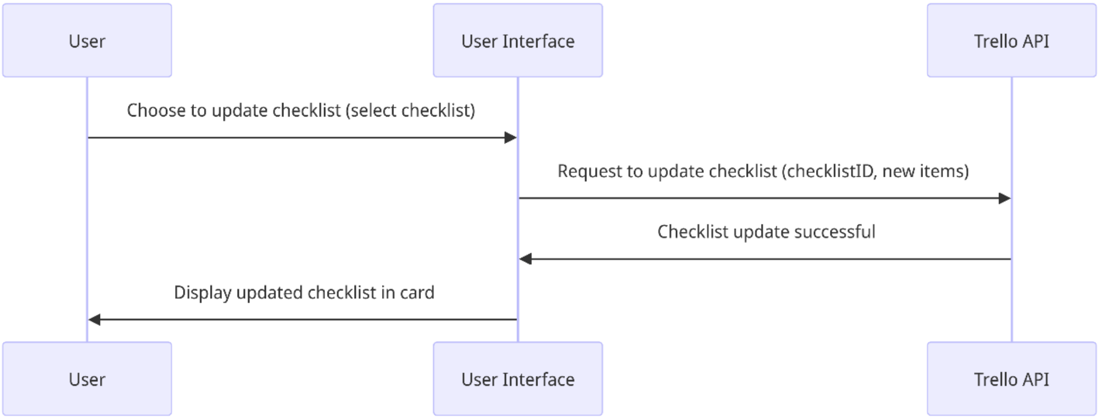
Joining a Card


User
User Interface
Trello API
Request to join a card (select card)
Request to add user to card (cardID, userID)
User added to card successfully
Display card with user as member
User
User Interface
Trello API

Managing Checklists on a Card (Creating)


User
User Interface
Trello API
Choose to add checklist to card
Request to create checklist (cardID, checklist name)
Checklist creation successful
Display updated card with new checklist
User
User Interface
Trello API

Managing Checklists on a Card (Updating)


User
User Interface
Trello API
Choose to update checklist (select checklist)
Request to update checklist (checklistID, new items)
Checklist update successful
Display updated checklist in card
User
User Interface
Trello API

Components Life Cycle In The Trelltech Project Management App

The lifecycle of components, particularly Flutter widgets in the TRELLTECH Project Management App, is central to how the app manages state and renders its user interface efficiently. This understanding is crucial for optimizing app performance and ensuring a smooth user experience.

Widget Lifecycle Overview

Flutter's approach to UI development is unique because of its widget-centric architecture. Widgets in Flutter can be either stateful or stateless, depending on whether they maintain state over time. Here's how the lifecycle of these widgets plays out in the TRELLTECH app:

Initialization: For stateful widgets, this phase includes the creation of the widget and the initialization of its state. This is where the initState method comes into play. In TRELLTECH, this is crucial for initializing the state with data fetched from the Trello API, such as the list of workspaces or boards.

Widget Rebuilding: Widgets are rebuilt in response to state changes. In Flutter, this is often triggered by calling setState within stateful widgets, which signals the framework to redraw the widget. In TRELLTECH, state changes occur due to user interactions, such as adding a new card to a list or updating a board's name. The efficient handling of these state changes is key to ensuring that the UI remains responsive.

Updating: When a widget's configuration changes, the framework calls the didUpdateWidget method of the widget's state class. This might happen if parent widgets pass down new data. In the context of TRELLTECH, this could involve updating a list when a new card is added to a board in a different part of the app, necessitating a widget update to reflect this new state.

Deactivation and Disposal: When a widget is removed from the widget tree, it goes through deactivation (deactivate method) and disposal (dispose method). This lifecycle stage is critical for resource management, ensuring that any listeners or controllers are properly disposed of to prevent memory leaks. In TRELLTECH, this could relate to disposing of controllers used for text input when a form widget is removed from the screen.

State Management and Transitions

State management is a cornerstone of the TRELLTECH app's functionality, ensuring that the UI accurately reflects the app's current state. We use a combination of Flutter's built-in state management capabilities and external packages for more complex scenarios:

Provider Package: For managing app-wide state, such as the current user's workspace selection, TRELLTECH utilizes the Provider package. This facilitates state sharing across different parts of the app without directly coupling widgets together, allowing for more readable and maintainable code.

Local State Management: For local widget state, such as the toggle state of a dropdown menu, TRELLTECH relies on the widget's internal state management mechanisms (setState). This approach is sufficient for simple state changes that don't need to be shared across multiple widgets.

Best Practices in TRELLTECH's Component Lifecycle Management

Lazy Loading: Widgets are designed to lazily load data as needed, minimizing initial load times and enhancing the user experience. Data fetching is triggered in the initState method but performed asynchronously to prevent UI blocking.

Memoization: To optimize performance, especially for expensive operations like filtering and sorting lists, TRELLTECH implements memoization techniques. This ensures that results from functions are cached when inputs are identical, reducing the need for re-computation.

Disposal of Resources: Ensuring timely disposal of controllers and listeners in the dispose method to prevent memory leaks and other performance issues.

Through meticulous management of component lifecycles and state transitions, TRELLTECH ensures a robust, efficient, and responsive project management application that stands up to the demands of its users while providing a seamless experience.

Contributors

This project is an academic endeavor by students at Epitech. For inquiries, contributions, or further information, please feel free to reach out to us via email.

Paternus DJEUKOUA TAMEN - paternus.djeukouatamen@epitech.eu

Adrien GARNIER - adrien.garnier@epitech.eu

Samir Sergio Boladji MIGUEL - samirsergio.miguel@epitech.eu

Your contributions and feedback are highly appreciated. For any technical issues or project-related questions, do not hesitate to contact us through our provided email addresses.