SOFTWARE ENGINEERING 2

travlendar+

Guglielmo Menchetti
Lorenzo Norcini
Tommaso Scarlatti

OVERVIEW

1. Requirements Analysis
2. System Design
3. Implementation
4. Testing
5. Demo Video

PURPOSE

- **Travlendar+** is a calendar based application to help users organise their appointments taking into account their preferences and providing appropriate mobility solutions.

- **Main goals and functionalities**
  - Manage events (standard, flexible, recurrent)
  - Guarantee schedule feasibility
  - Provide mobility solutions according to preferences
  - Offer booking functionalities

REQUIREMENTS ANALYSIS

GOALS (8)

- **[G.1]** Access from different platforms
- **[G.2]** Manage meetings
- **[G.3]** Reach every meeting on time
- **[G.4]** Select or edit a travel mean to reach an event
- **[G.5]** Set preferences
- **[G.6]** Create flexible events and recurrent events
- **[G.7]** Book transportation for a trip
- **[G.8]** Be notified before the occurrence of an event

CONSTRAINTS AND DEPENDENCIES

- **Location:** Milan
- **Platform:** compatible OS or browser
- **Connection:** Internet connection
- **External services:** third-party APIs
  - Travel options
  - Booking
- **Persistent data:** a DBMS to store/retrieve data

DOMAIN ASSUMPTIONS (12)

- **Data correctness**
  - [D.3] Events informations provided by the user are correct
- **Network resiliency**
  - [D.1] The sent email is assumed to be correctly received
- **Travel availability**
  - [D.9] All selected travel means specified in the preferences are available
- **Booking service**
  - [D.11] User is registered to the service which offers the booking option.

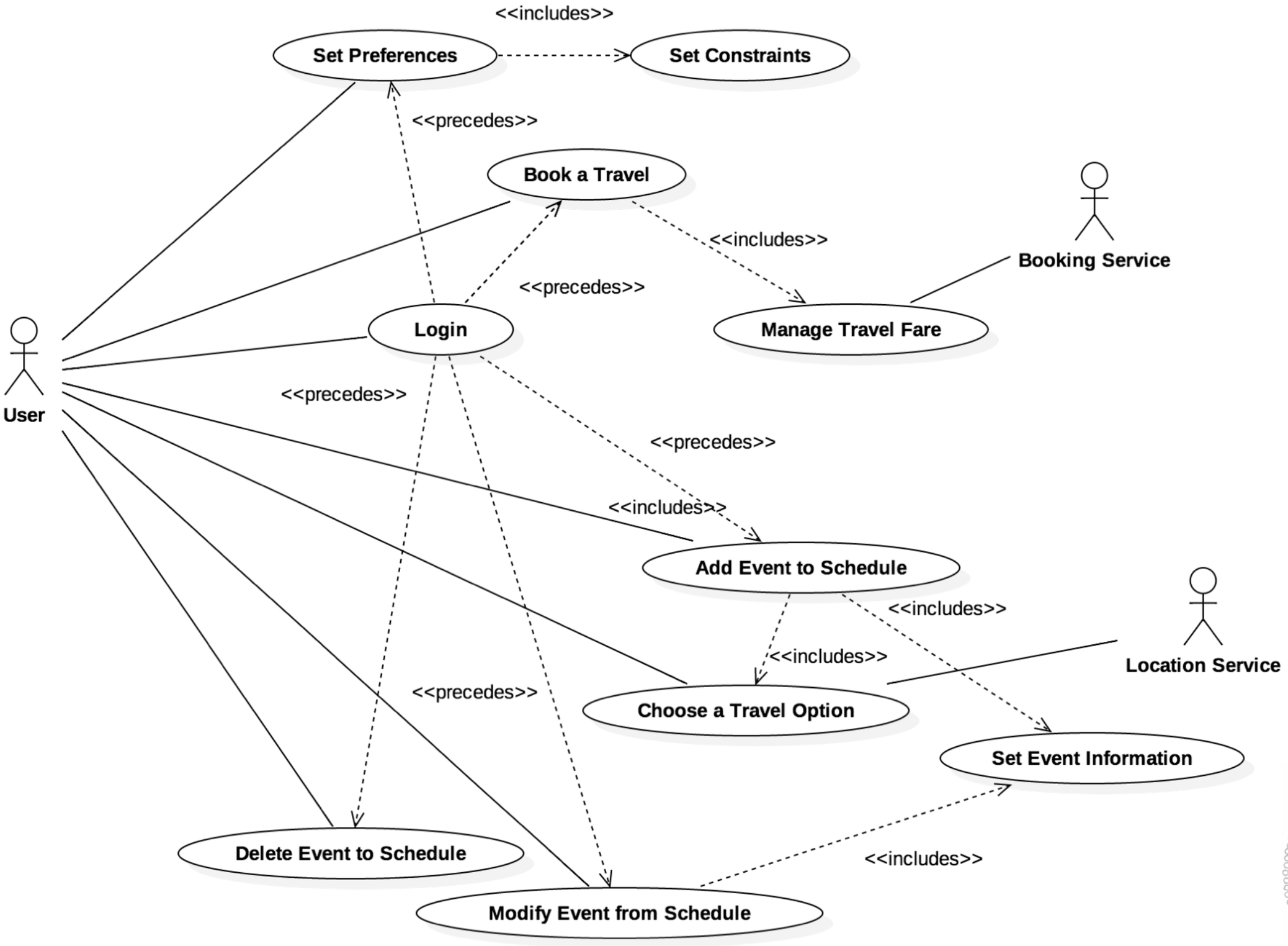
WORLD AND MACHINE PARADIGM

World phenomena

Machine phenomena

Shared phenomena

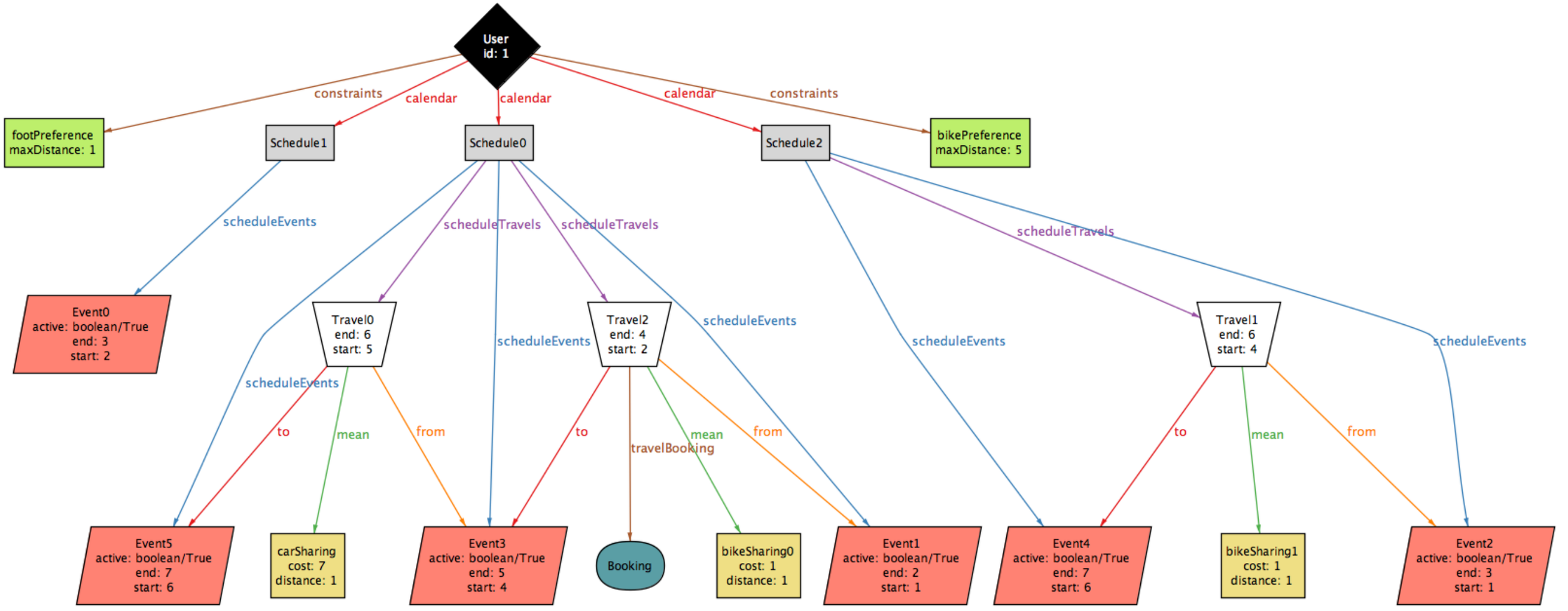
USE CASE DIAGRAM



FUNCTIONAL REQUIREMENTS (29)

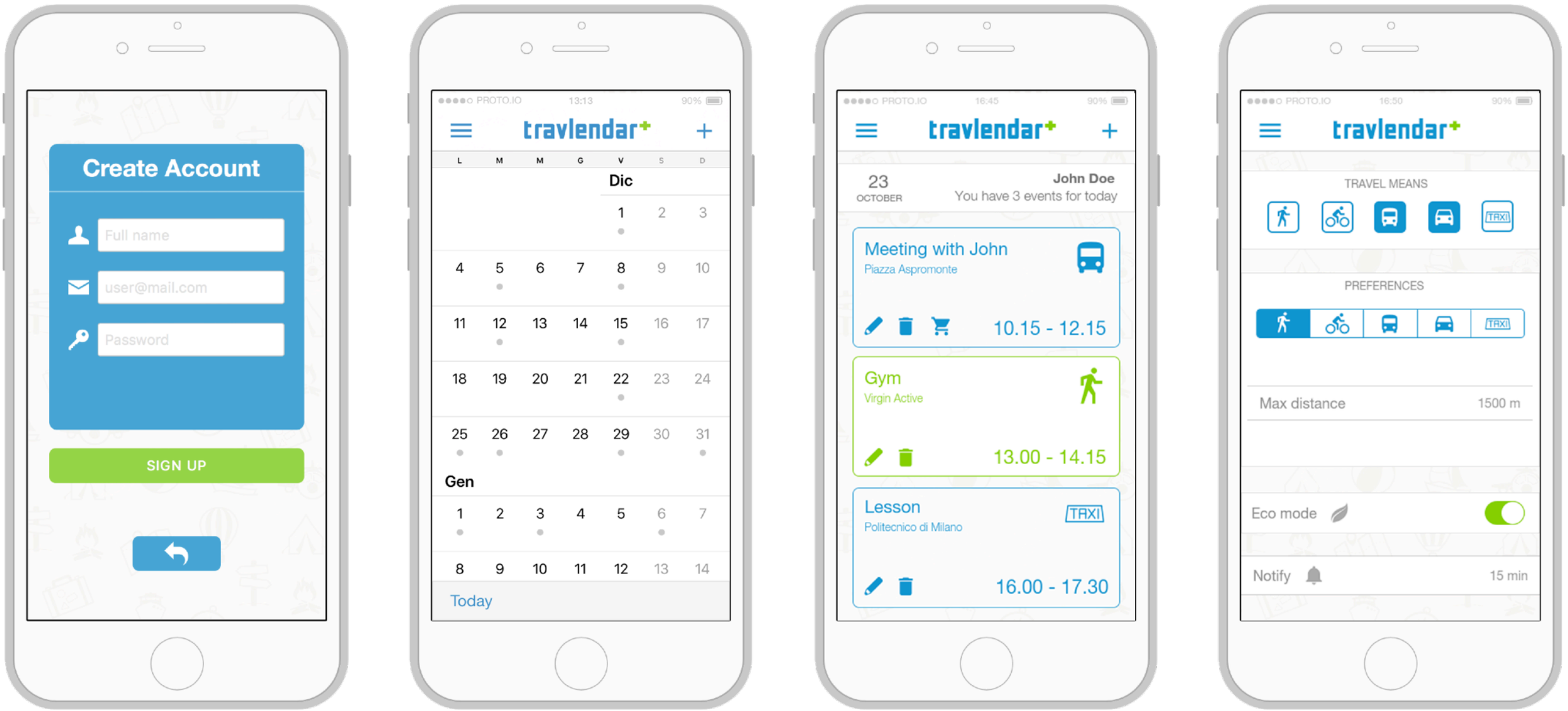
- **User registration/login**
  - [R.4] Allow the User to log in using his/her personal credentials
- **Event management**
  - [R.7] The User must be allowed to create events, specifying…
  - [R.11] Check if the event created or edited by the User is feasible.
- **User preferences/constraints**
  - [R.18] User can define specific constraint for each travel means, that are…
- **Third party APIs**
  - [R.27] Provide interface for third party services allowing the User to authenticate
- **Notifications**
  - [R.28] Allow to activate notifications and setting their time

ALLOY MODEL



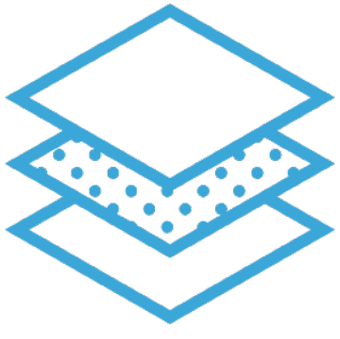
- Users: **1**
- Schedules: **3**
- Events: **6**
- Travels: **3**
- Booking: **1**

MOCKUPS



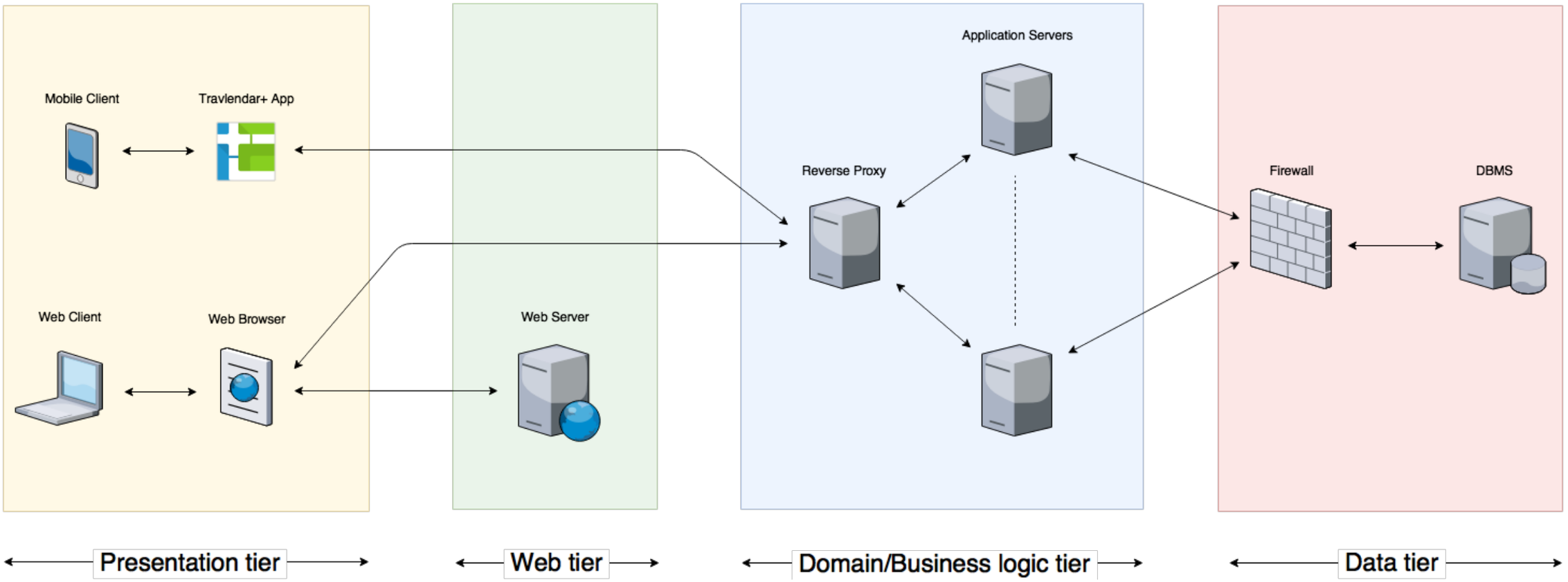
- Application prototyping platform: **Proto.io**

SYSTEM DESIGN

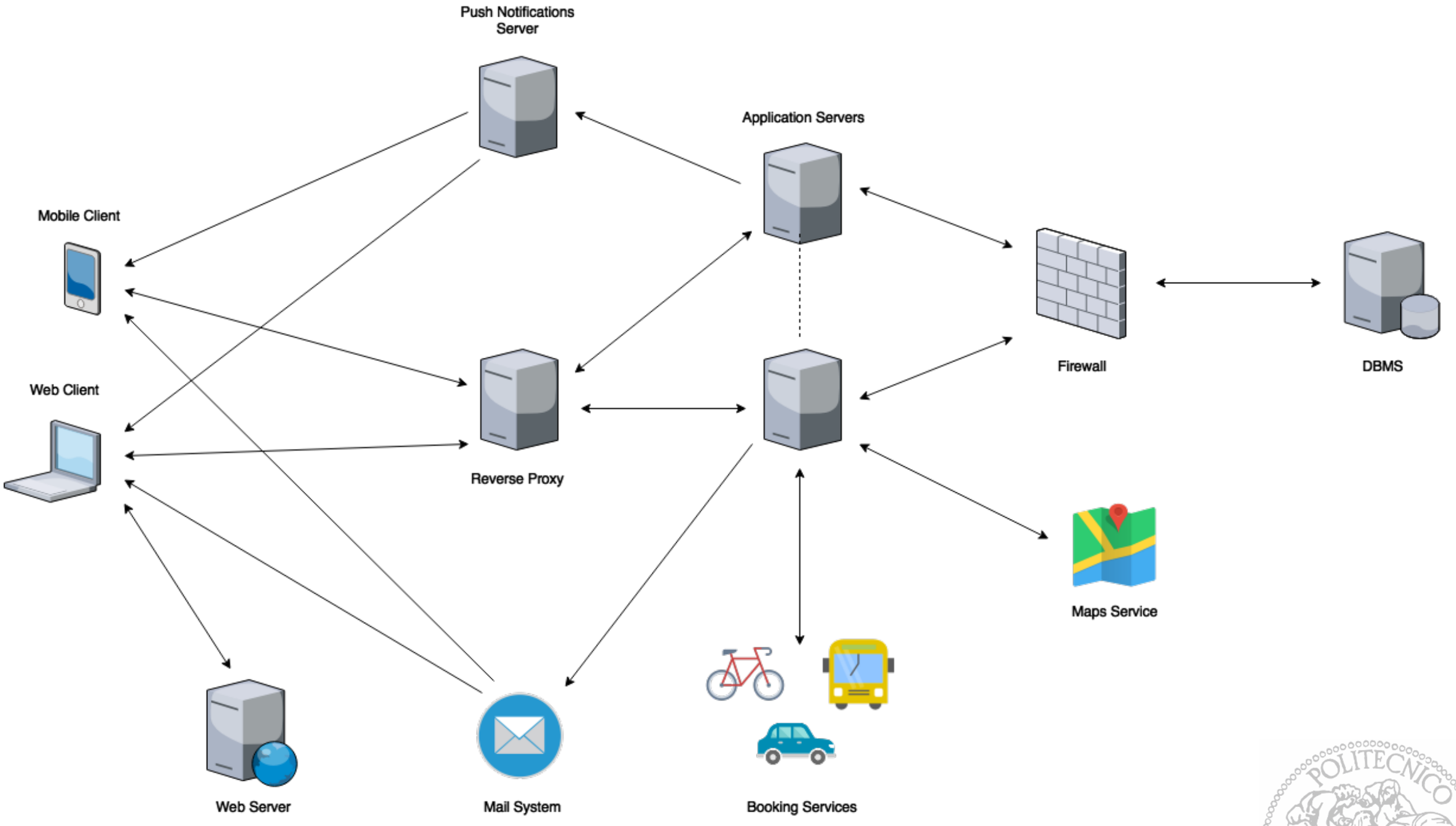


SYSTEM ARCHITECTURE

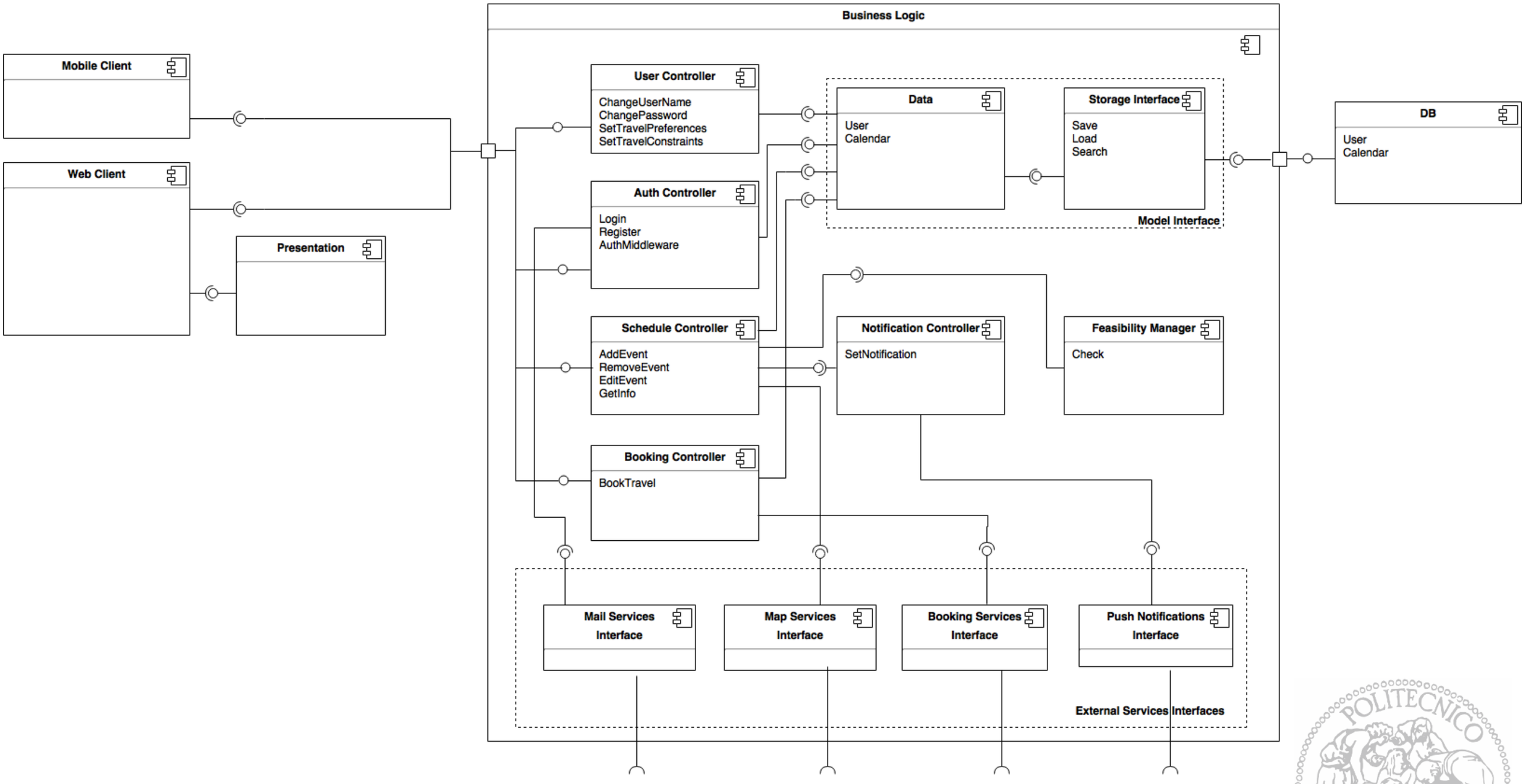
- **Multitier Architecture**
  - *Presentation Tier*: interface the User interacts with
  - *Web Tier*: provides client side application
  - *Business Logic Tier*: controls application's functionality
  - *Data Storage Tier*: used to store data



SYSTEM OVERVIEW



COMPONENT DIAGRAM

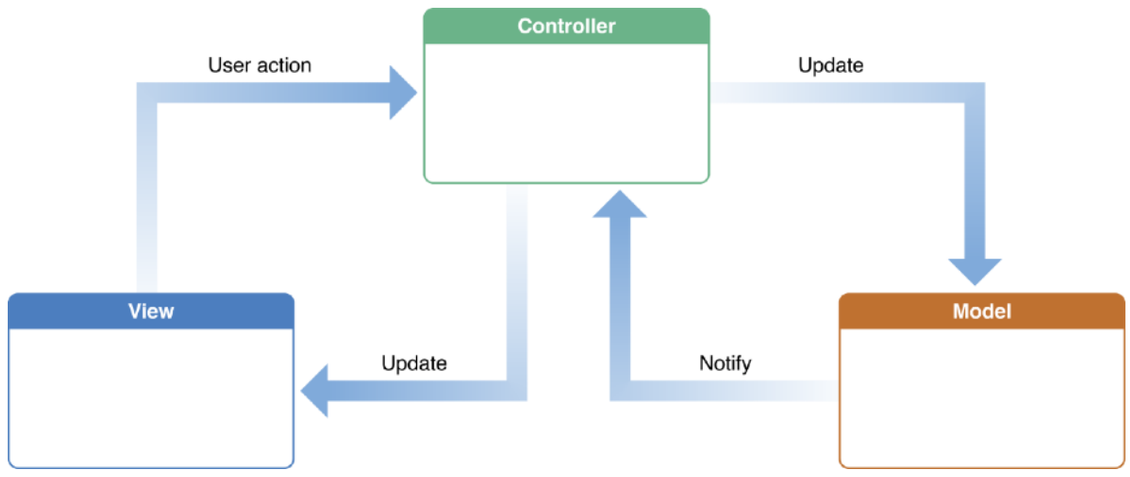


REQUIREMENTS TRACEABILITY

Component	Requirements
Auth Controller	• [R.1] A Guest must be able to register. During the registration the System will ask to provide credentials. • [R.2] Check if the Guest credentials are valid. • [R.4] Allow the User to log in using credentials.
Feasibility Manager	• [R.11] The System must check if the event created or edited by the User is feasible. • [R.14] The System must guarantee a feasible schedule, that is, the User is able to move from an appointment to another in time.
• • •	

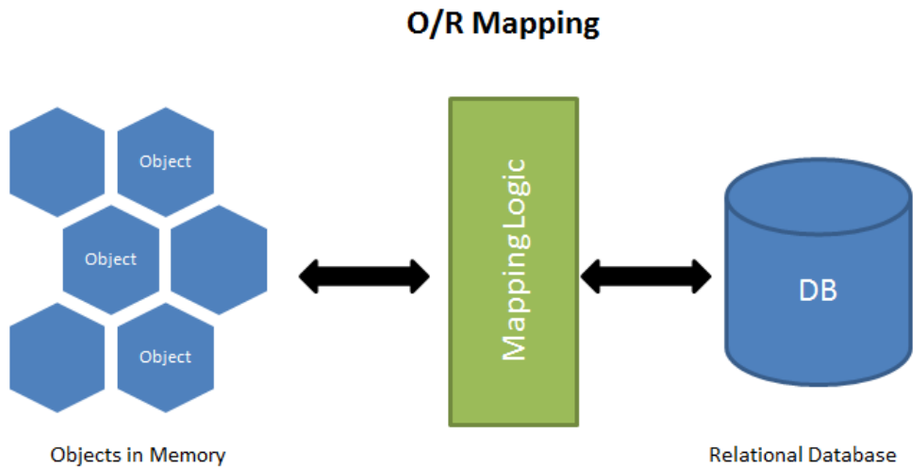
DESIGN CHOICES AND PATTERNS

- **Thin Client**
  - Close to no computation
  - Handles communications
  - Easy data synchronization
  - Less effort for implementation in different Clients
- **Model View Controller**
  - Change of components without hassle
  - I&T without fully implemented components



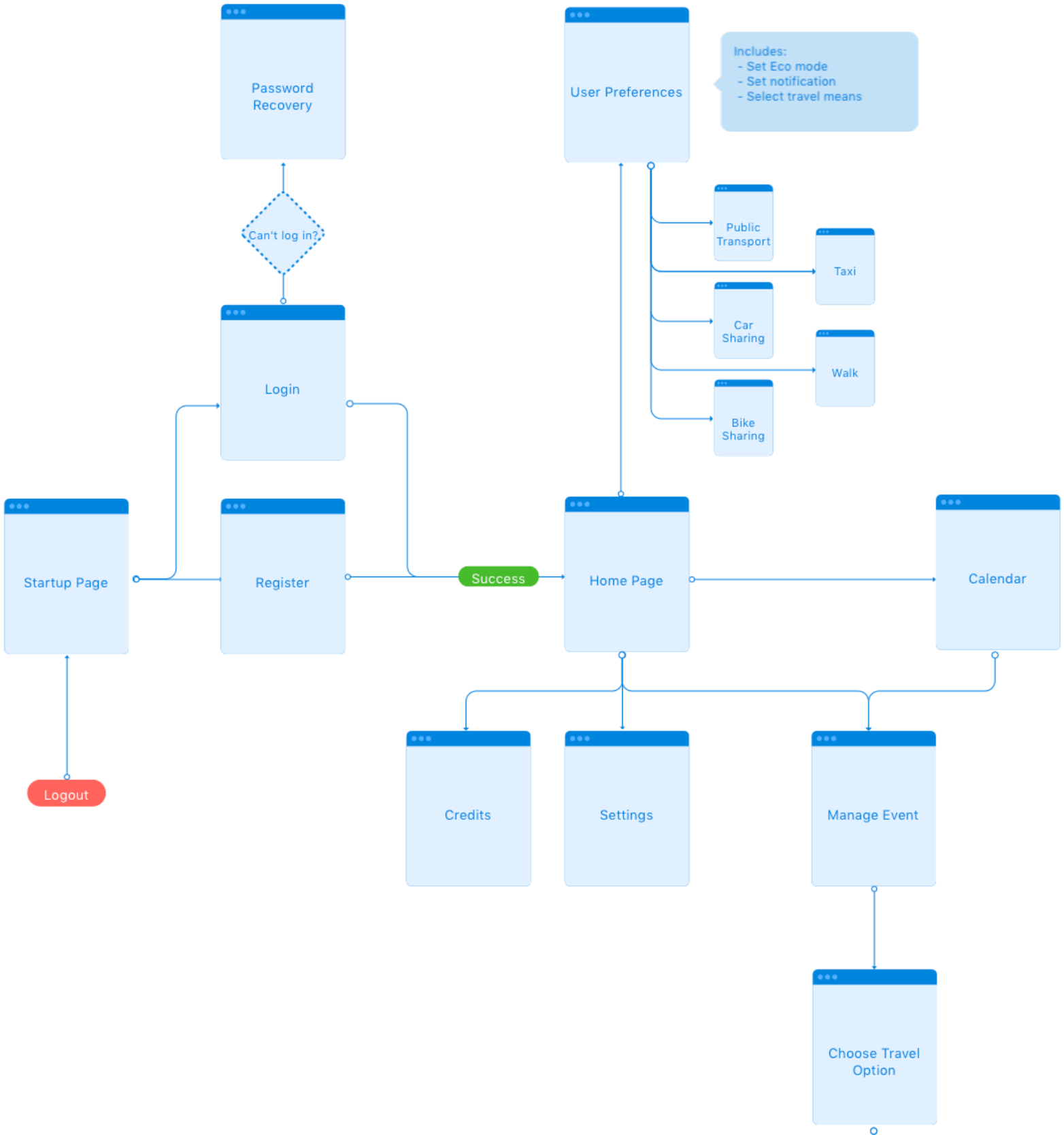
DESIGN CHOICES AND PATTERNS

- **Object-Relational Mapping**
  - Query and manipulate data using object-oriented paradigm
  - More reusable and cleaner code



- **RESTful**
  - Data exchange through HTTP protocol
  - Stateless: requests contain all the necessary information
  - Uniform Interface: enhance scalability

UX DIAGRAM

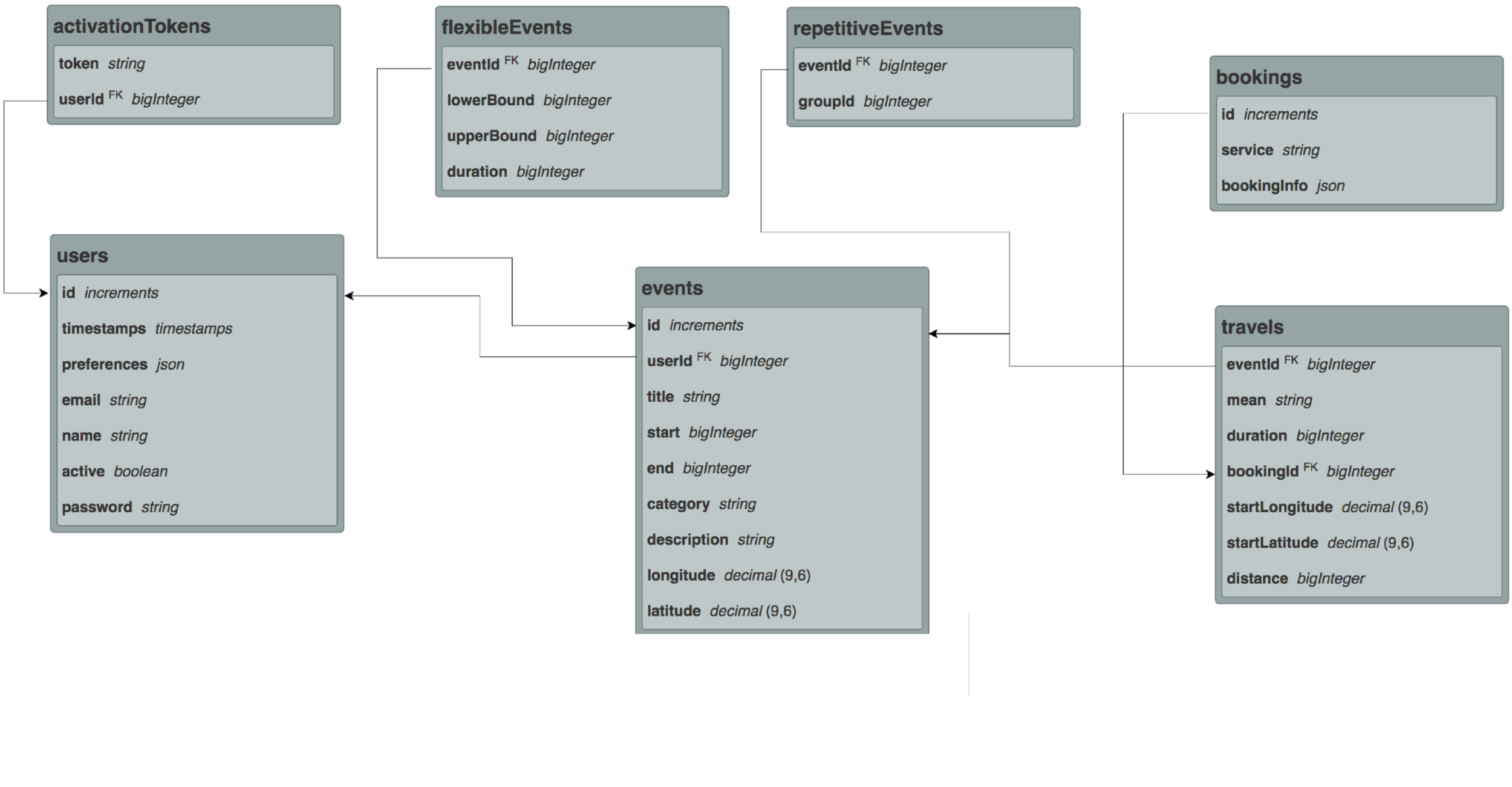


- **User flow:** path the user follows through the application

IMPLEMENTATION DETAILS

DATABASE SCHEMA

- **DBMS:** Relational DBMS (PostgreSQL 9.6)



BACKEND DETAILS

- **Framework:** Laravel 5.4
- **Language:** PHP 7.1
- **DBMS Interface:** Eloquent (ORM)
- **Authentication:** Passport (Token)
- **APIs:** RESTful

BACKEND FUNCTIONALITIES

- **User**
  - User credentials
  - Preferences
- **Schedule**
  - Constraint Satisfaction Problem to solve feasibility
  - Flexible events adjustment
  - Recurrent events as single events after creation
- **Travel**
  - Accounting for travel time
  - Adapting schedule for each option

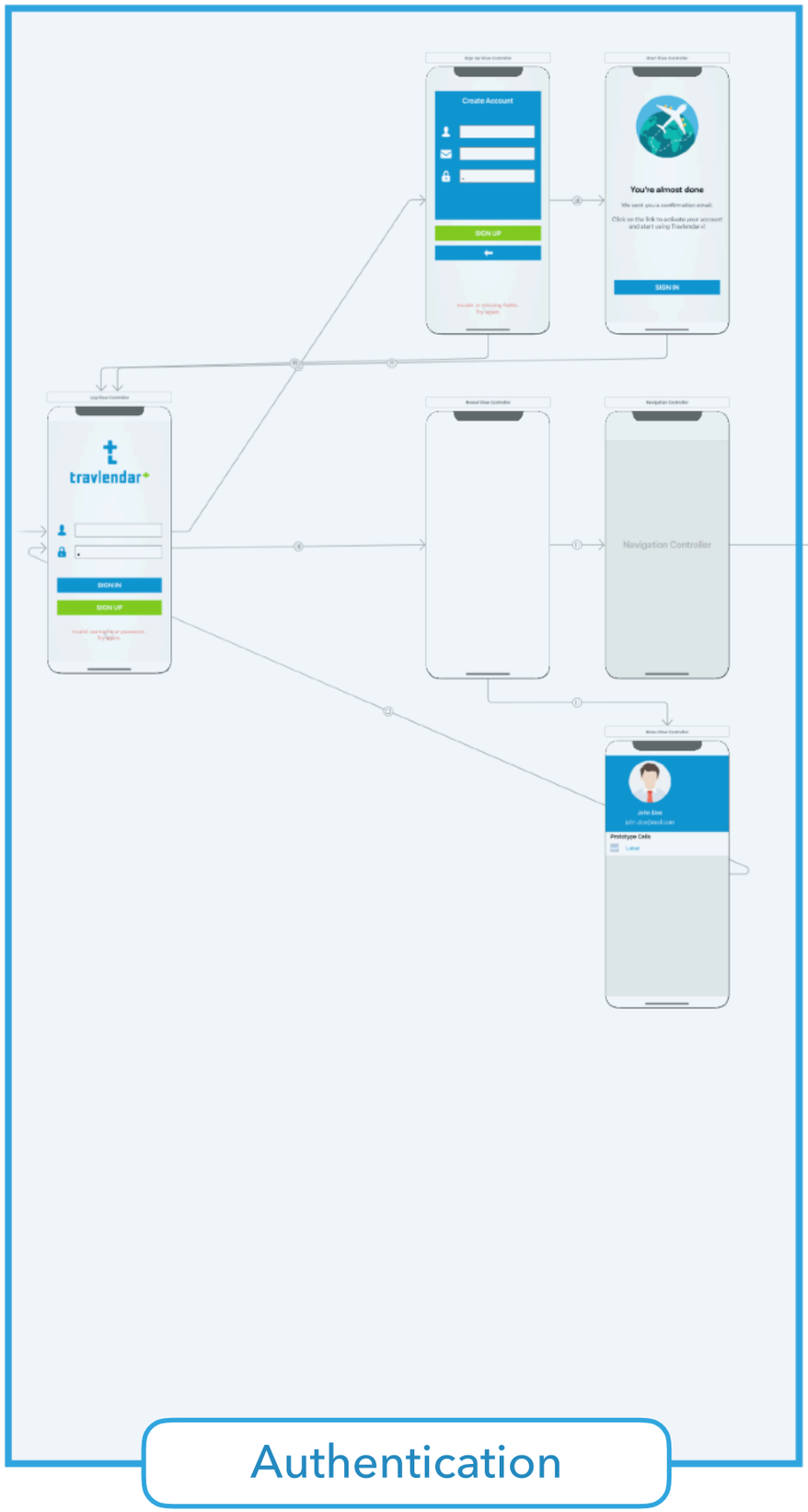
EXTERNAL SERVICES

- **Travel APIs**
  - Google Directions: public transport, personal transport and foot
  - Mapbox: bicycle
  - Uber: available services
- **Booking APIs**
  - Uber: book available service
- **Mail Service**
  - Google Mail as mail server

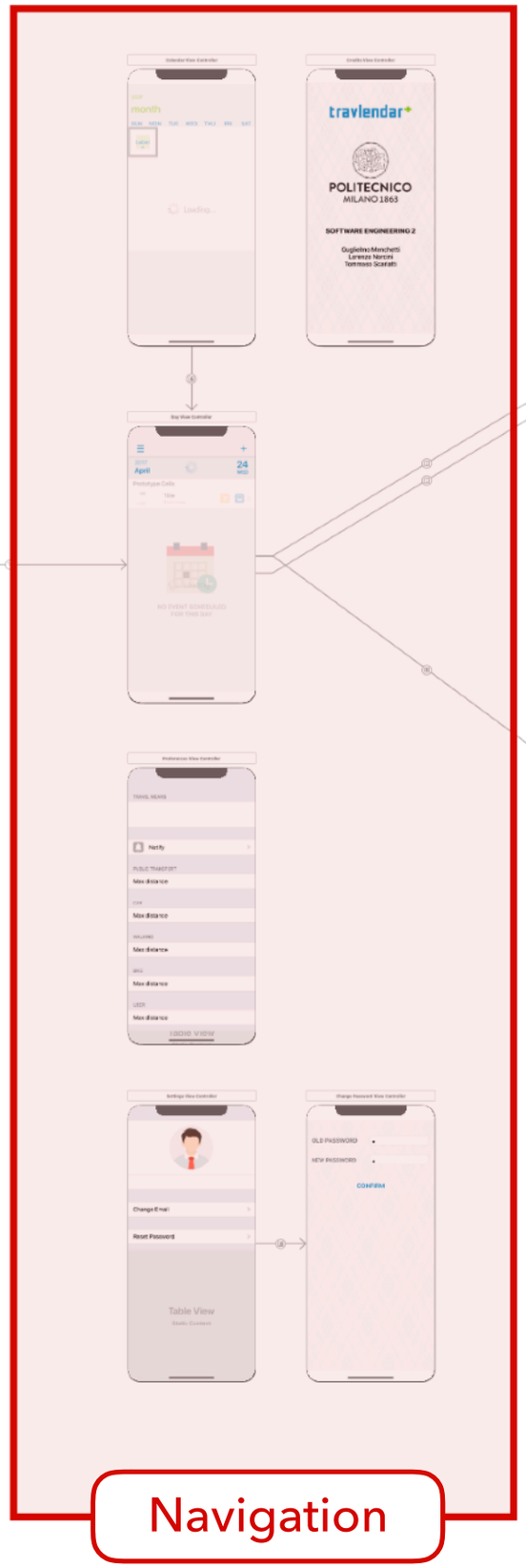
FRONT END DETAILS

- **Target:** iOS 11
- **Language:** Swift 4
- **Frameworks and SDKs**
  - *JTAppleCalendar*: build a calendar from scratch
  - *SwiftDate:* manage dates and timezones in Swift
  - *Alamofire:* make elegant HTTP requests
  - *UberRides:* integrate Uber Rides API

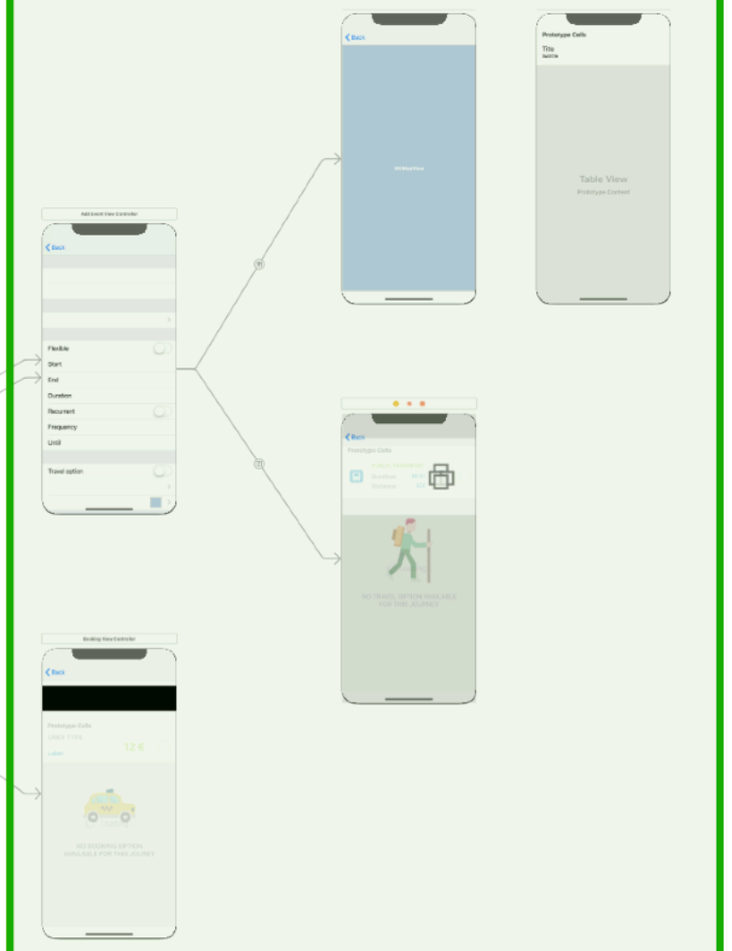
FRONT END STORYBOARD



Authentication



Navigation

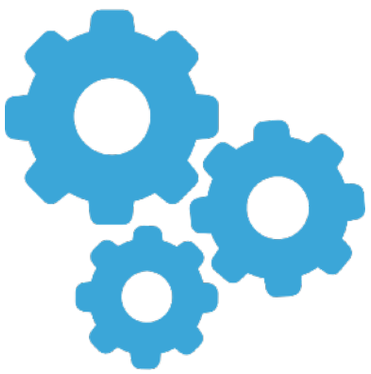


Event-related

IMPLEMENTED FUNCTIONALITIES

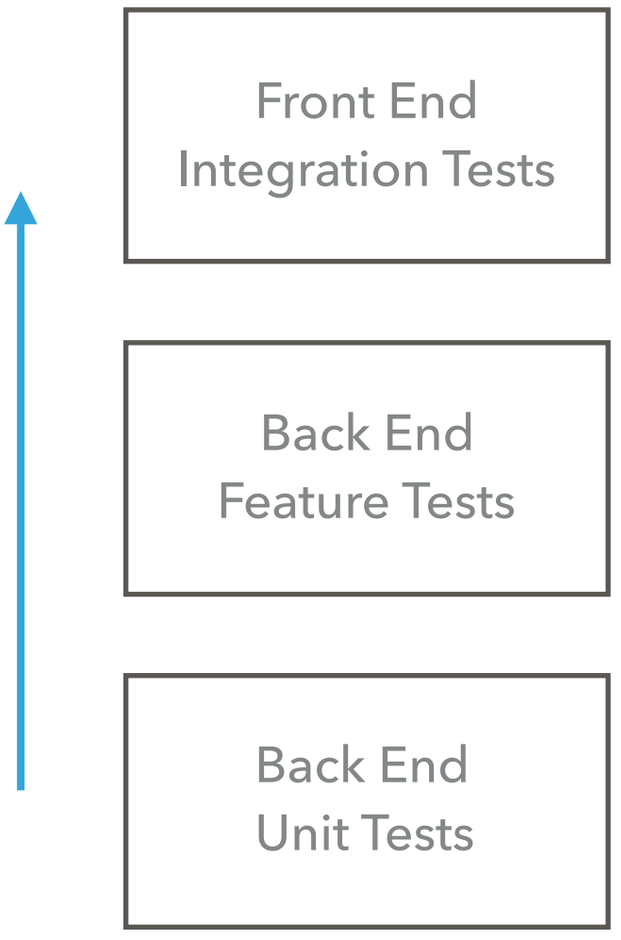
- **[G.1]** Access from different platforms
- **[G.2]** Manage meetings
- **[G.3]** Reach every meeting on time
- **[G.4]** Select or edit a travel mean to reach an event
- **[G.5]** Set preferences
- **[G.6]** Create flexible events and recurrent events
- **[G.7]** Book transportation for a trip
- **[G.8]** Be notified before the occurrence of an event

TESTING



TESTING OVERVIEW

- **Bottom Up Approach**
  - Unit test of backend components
  - Feature test of APIs
  - Integration test of frontend and backend
- **Testing Frameworks**
  - PHPUnit
  - XCTest



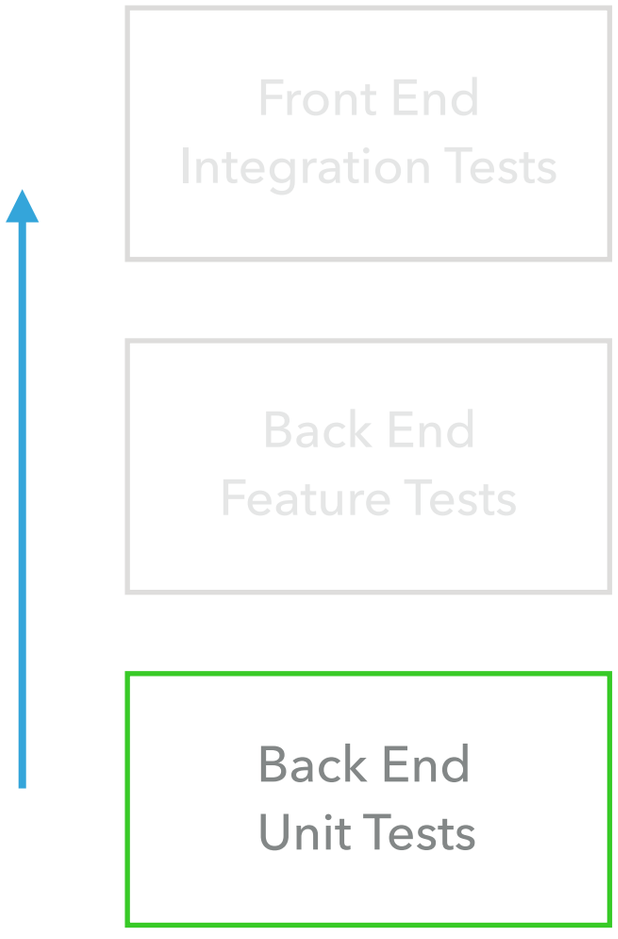
BACKEND UNIT TESTS

Tested Components

- Booking Interface
- Maps Interface
- Constraint Satisfaction Problem solver
- Feasibility Manager

Performed Checks

- Functions return values



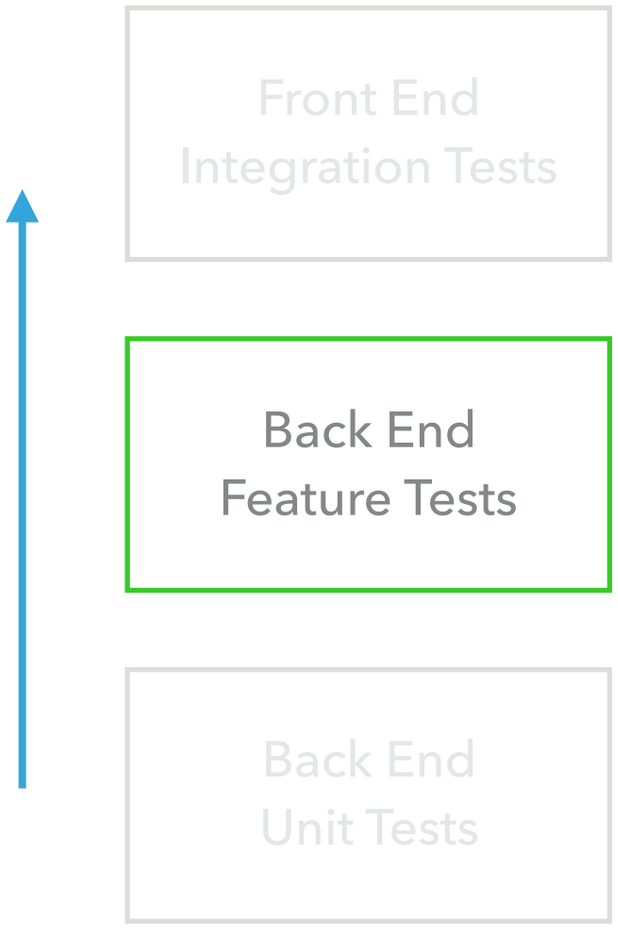
BACKEND FEATURE TESTS

Tested APIs

- Authentication
- User Management
- Event Management
- Booking
- Travel

Performed Checks

- HTTP response code
- HTTP response body
- Data storage and retrieval



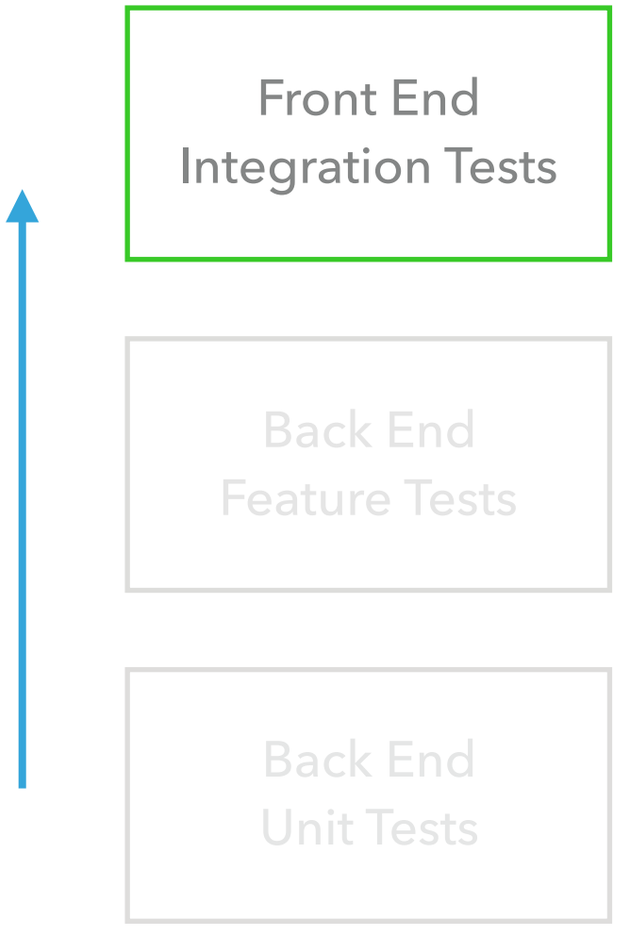
FRONTEND INTEGRATION TESTS

Tested Functionalities

- Authentication
- User Management
- Event Management
- Booking
- Travel

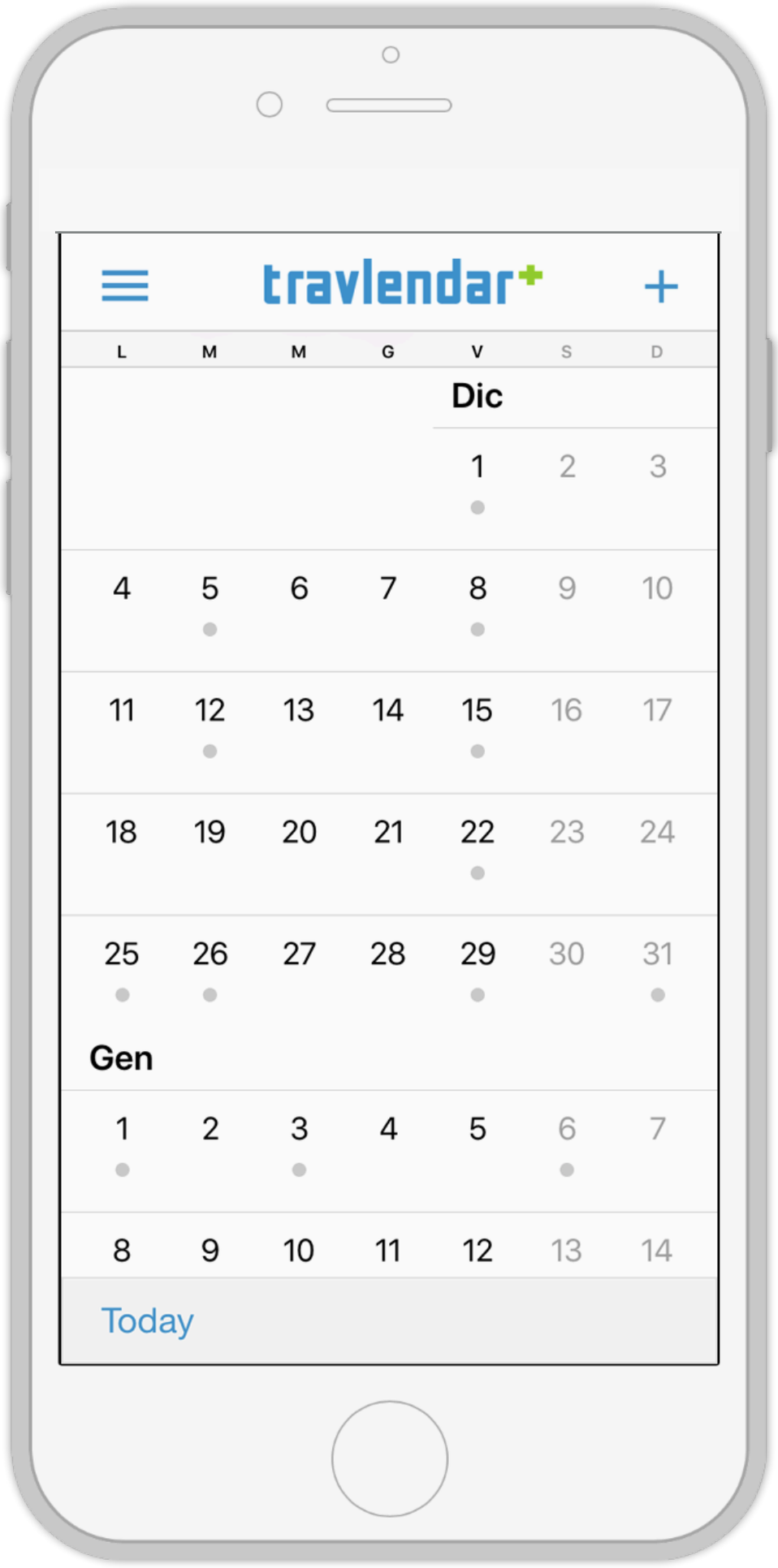
Performed Checks

- HTTP response code
- Effects of requests on the system



DEMO
VIDEO

- SIGN UP
- CONFIRMATION EMAIL
- LOG IN
- ADD STANDARD EVENT
- ADD RECURRENT EVENT
- ADD “LUNCH” EVENT
- AUTOMATIC ADJUSTMENT
- SET PREFERENCES
- ADD EVENT WITH TRAVEL
- BOOK RIDE WITH UBER
- DELETE EVENT
- SETTINGS
- CREDITS
- LOGOUT



SOFTWARE ENGINEERING 2

THANK YOU FOR YOUR ATTENTION

Guglielmo Menchetti
Lorenzo Norcini
Tommaso Scarlatti