Olympia Dictionary

Capstone Project Design Specification

Sergii Opryshko 7759145 sopryshko9145@conestogac.on.ca
Brahmpreet Singh 7845159 brahmpreet.official@gmail.com
Sharfaraz Ahamed Bachinvalapan Hameed 7834120 sharfrazq17@gmail.com
Anju Mathew 7804396 10mathew10@gmail.com

School of Engineering & Information Technology
Conestoga College Institute of Technology and Advanced Learning
August 1, 2018

Software requirements

The following requirements were agreed with the client and implemented:

SR 1. User

- SR 1.1. Register a new user on a web service
- SR 1.2. Log in on a web service
- SR 1.3. Reset password
- SR 1.4. Validate input fields
- SR 1.5. Change first name and last name
- SR 1.6. Change email
- SR 1.7. Change password
- SR 1.8. Logout in the app
- SR 1.9. Delete account

SR 2. Words vocabulary

- SR 2.1. Search a word in a dictionary. Show a word card
- SR 2.2. Add the word to the list of previously stored words
- SR 2.3. Remove a word from the list of previously searched words
- SR 2.4. Sort words in ascending/descending order (by name and date added)
- SR 2.5. Copy list of words to clipboard
- SR 2.6. Save vocabulary (words and their categories) to local database
- SR 2.7. Restore vocabulary (words and their categories) from local database
- SR 2.8. Delete all from local database

SR 3. Words categories

- SR 3.1. Create category
- SR 3.2. Remove category
- SR 3.3. Rename category
- SR 3.4. View categories
- SR 3.5. Assign a category to the word
- SR 3.6. Filter words list by category
- SR 3.7. Remove filtration by category

SR 4. UI

SR 4.1. Splash screen
SR 4.2. About page
SR 4.3. Legal page
SR 4.4. How-To alert window on start up

SR 5. Quizzes

SR 5.1. Sequence of cards with definitions for self-check
SR 5.2. Check the correctness of given answer for the definition
SR 5.3. Quiz results page
SR 5.4. Select category of words for the quiz
SR 5.5. Overall statistics for all the quizzes

SR 6. Settings

SR 6.1. Change UI language
SR 6.2. Change dictionary language (must use other APIs available for a particular language)
SR 6.3. Change input type to text
SR 6.4. Change input type to speech
SR 6.5. Change input type to OCR
SR 6.6. Detection of no internet connection. “Enable Wi-Fi or mobile data” dialog
SR 6.7. Change font size of UI
SR 6.8. Change colors for text and its background

SR 7. Accessibility

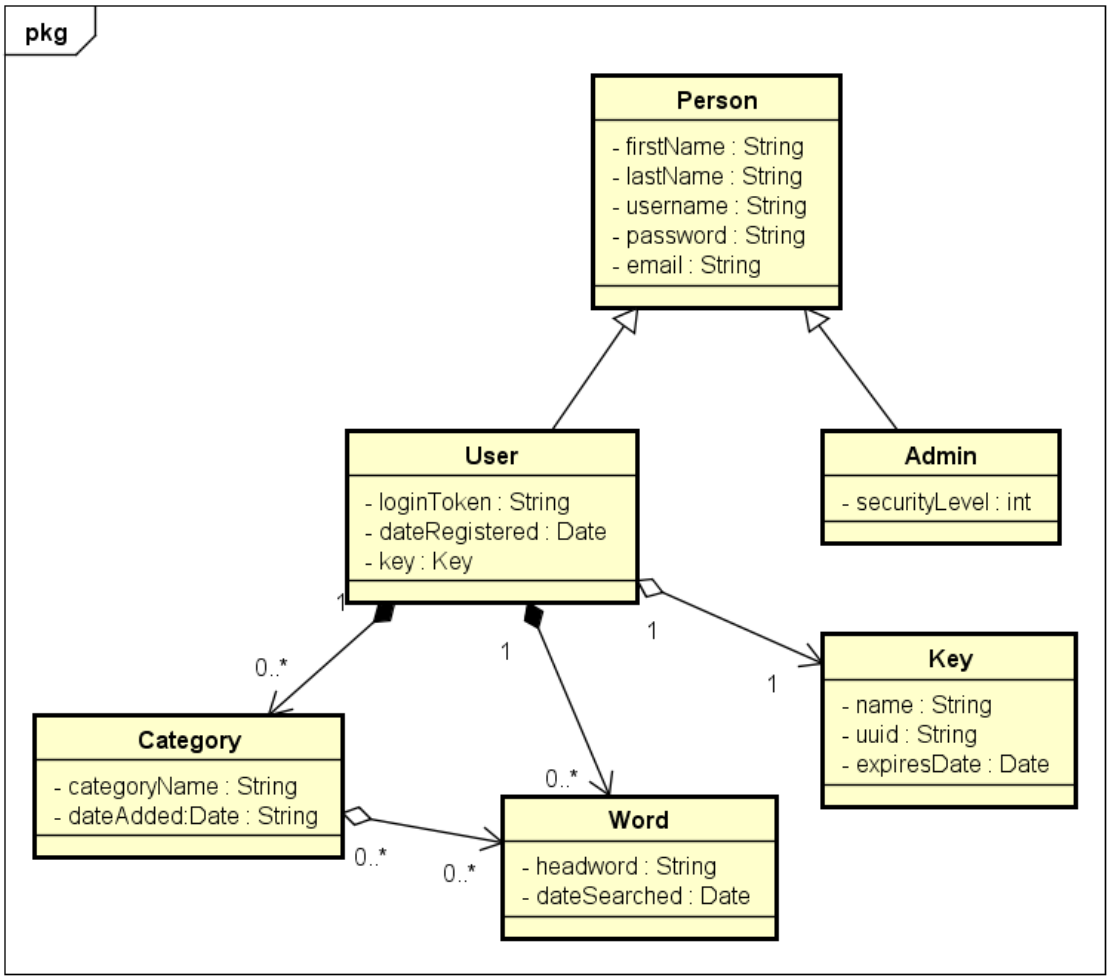
SR 7.1. Text-to-speech support for word cards
SR 7.2. Text-to-speech support for word definition in the quiz
SR 7.3. Speech-to-text support for answer field in the quiz
SR 7.4. Switch to high contrast theme

SR 8. Server side

SR 8.1. Add an Olympia key check on login (in the mobile app)
SR 8.2. Log in to admin portal (as administrator)
SR 8.3. List all the application keys
SR 8.4. Issue a new Olympia key
SR 8.5. Change the expiry date of the key, making it obsolete
SR 8.6. List all the users of the app registered in database
SR 8.7. Change Olympia app key for particular user or subset of users. Those with an obsolete key will not be able to log in and continue using the app
SR 8.8. List all the administrators
SR 8.9. Add another administrator
SR 8.10. Update administrator account’s information (owned or another)
SR 8.11. Delete administrator account (owned or another)

UML Class diagram

UML class diagram is simple and reflects the idea of user having words and categories. Categories may or may not contain words. Words may or may not belong to any number of categories. The diagram describes the system in general, both application and server parts.



UML Sequences

The idea of the architecture is to give the application only the role of content receiver. Thus, on every request to the server we need to check if the user (application) has a valid key to proceed. As keys may be changed at any time by the administrator, it is necessary to check it every time the application asks for something from the server.

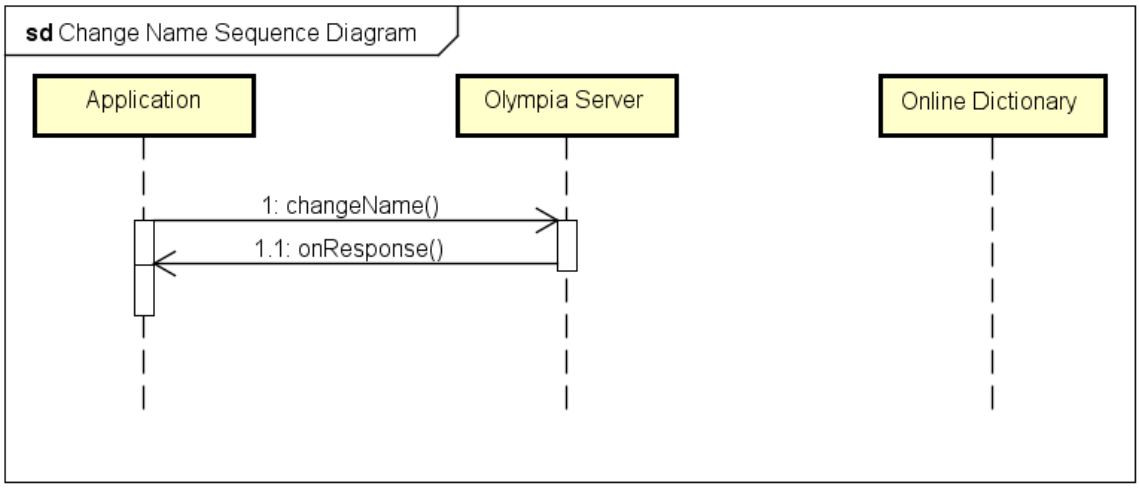
Only account related requests are permitted without a valid Olympia key. Upon registration a 5 days trial key is automatically assigned to the user and allows to try the service. All further requests require a key and check that it is not expired. The key can be purchased on a web site or directly from the phone.

The list of application requests:

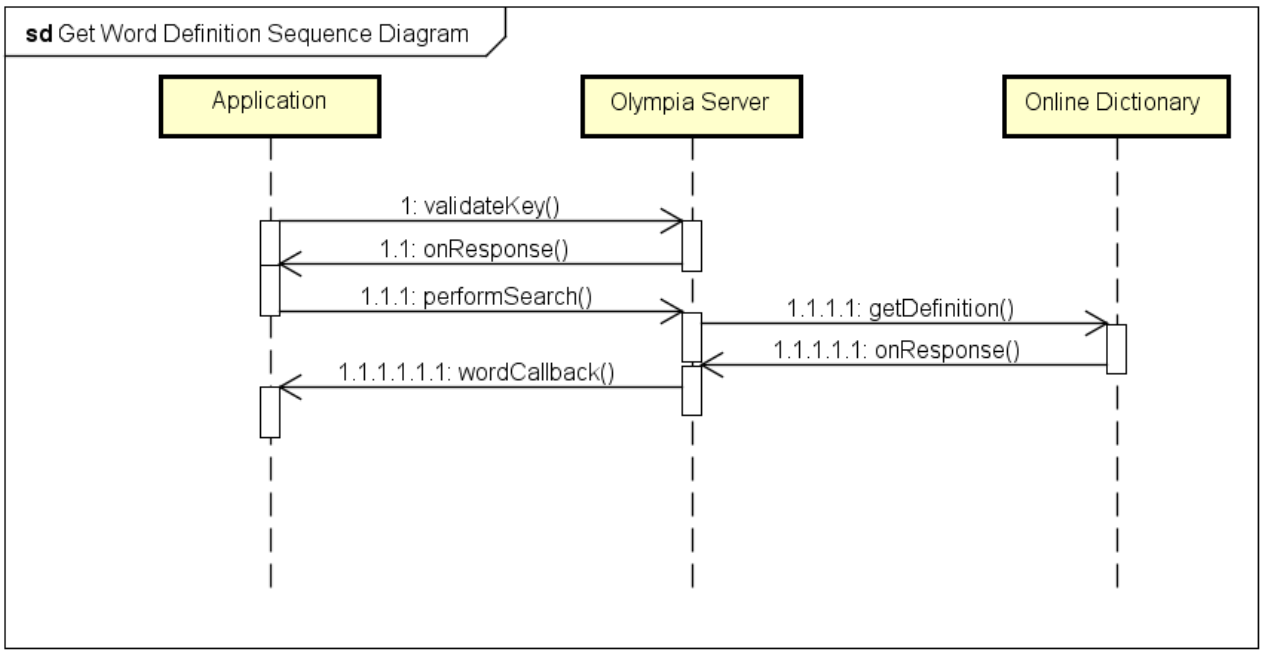
Request	Needs a key?
Register	No
Log in	No
Reset password	No
Verify password	No
Change name	No
Change email	No

Change password	No
Delete account	No
Get key	No
Get privacy policy	No
Get terms and conditions	No
Get word definition	Yes
Get synonyms	Yes
Get antonyms	Yes
Get inflections	Yes
Get sentences examples	Yes

Schema for requests that do not require a valid Olympia key:



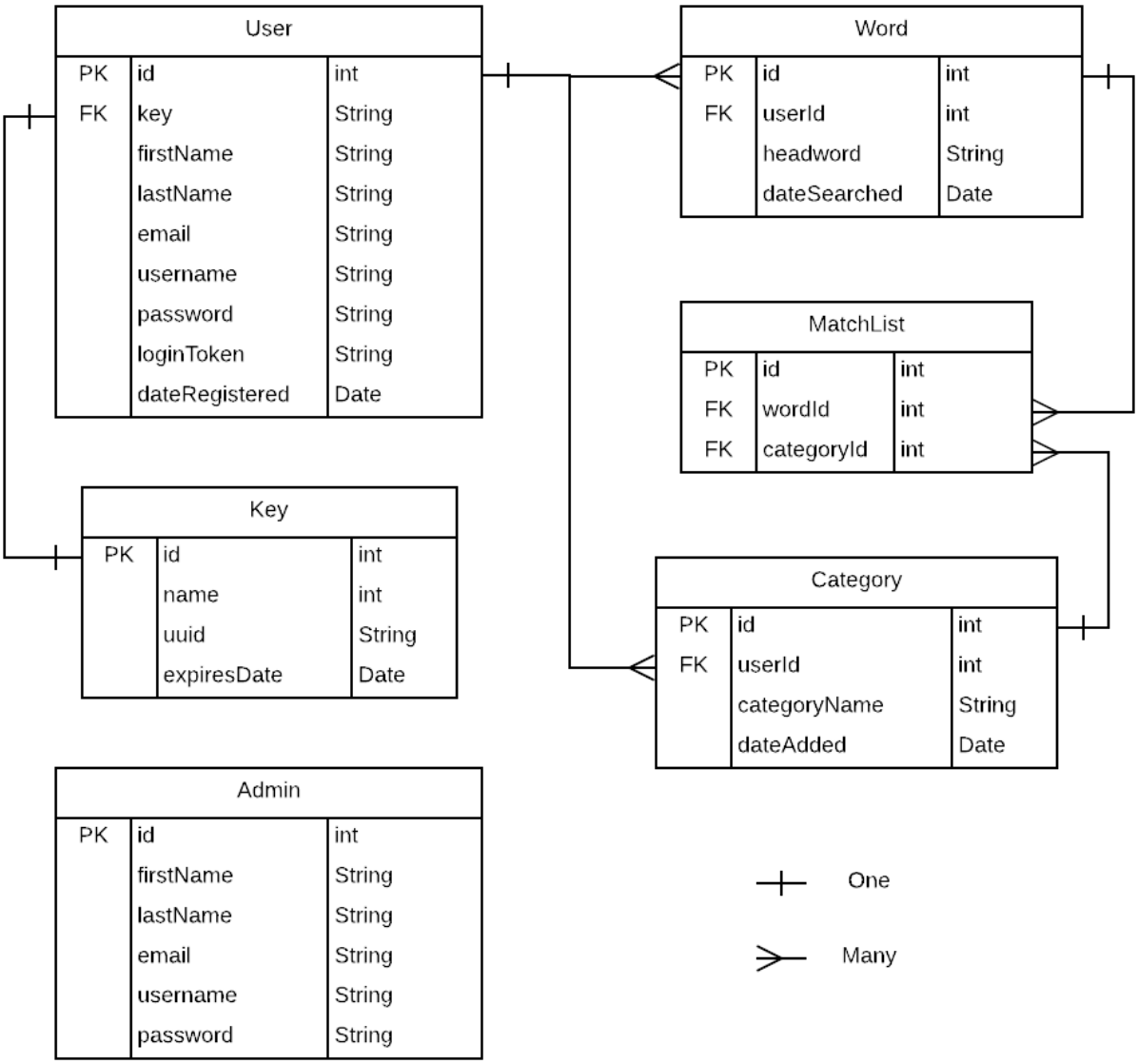
Schema for requests that do require a valid Olympia key:



Entities relationship diagram

A MongoDB NoSQL database was chosen because of better scalability and flexibility than its relational counterpart. Unlike relational models, which require predefined schema, NoSQL databases offer flexible schema design that make it much easier to update the database to handle changing application requirements. It is also convenient to store unstructured data in it and scale it horizontally when the need arises. On top of that, it is open source, which means we do not have to pay any software licensing fees.

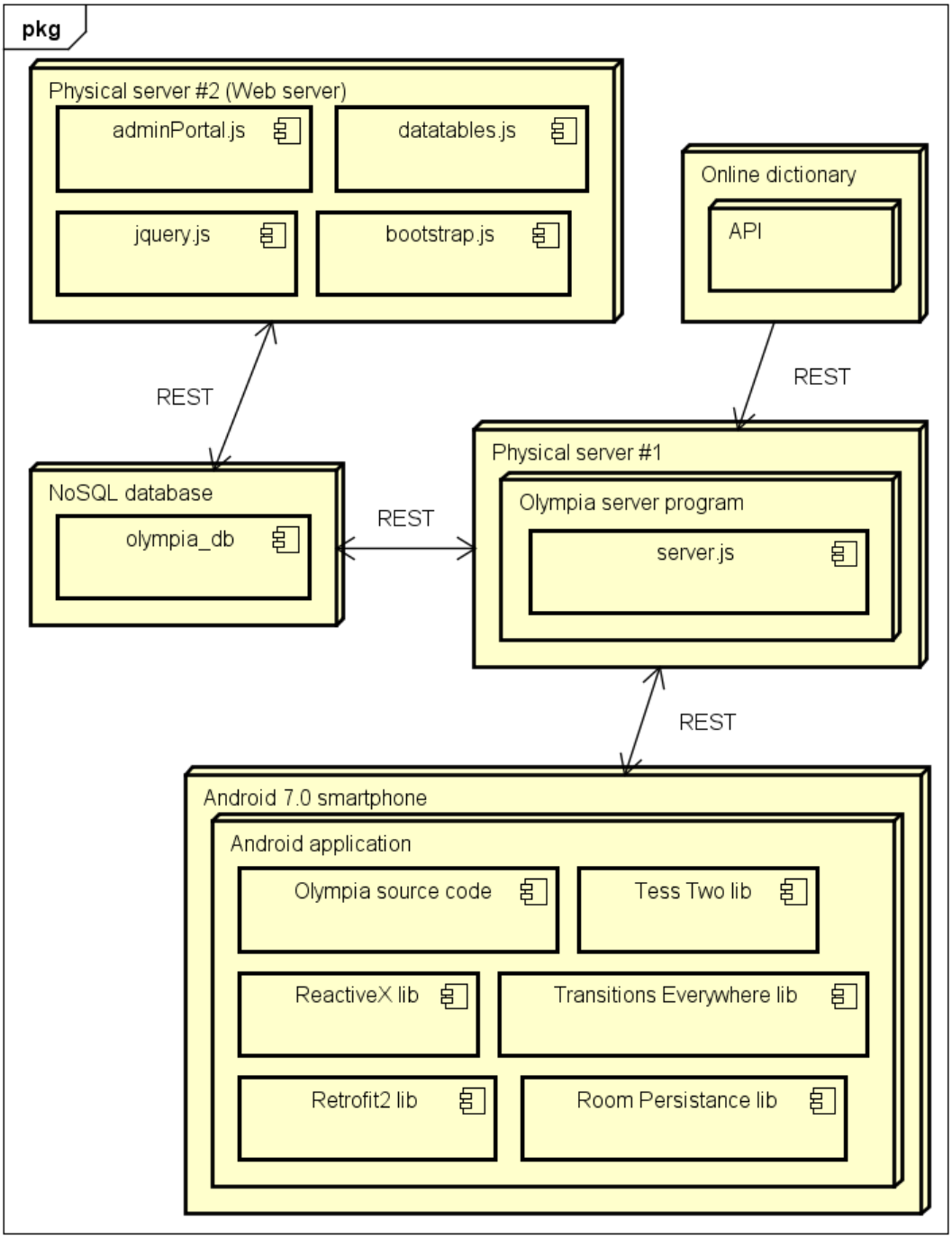
The database is connected to a NodeJS server via a plugin and contains the following data:



Admin and user tables are separate as administrators' records are only used on the server, while users' data is used on both server and smartphone. Key table is for Olympia application keys, while the dictionary API keys obtained from third parties are not stored in the database.

Deployment diagram

Smartphone app is distributed via Google Play store. Download is free. Olympia server is hosted on a datacenter server with firewall and DDoS protection. Web server can be hosted on the same or different physical machine. Both server and admin panel use the database located on the third server. For all three parts it's possible to rent capacities and deploy them there. System uses public APIs from online dictionaries hosted elsewhere.



System is flexible and scalable horizontally. More servers and a load balancer can be added to handle exceeding requests from users.

Libraries used

Brief description of libraries used in both server and application parts:

1. **Retrofit** is a type-safe HTTP client for Android and Java. It makes it easy to connect to a REST web service by translating the API into Java interfaces. Retrofit2 usage - https://code.tutsplus.com/tutorials/sending-data-with-retrofit-2-http-client-for-android--cms-27845
2. The **Room** persistence library provides an abstraction layer over SQLite to allow for more robust database access while harnessing the full power of SQLite. Room database official documentation - https://developer.android.com/topic/libraries/architecture/room
3. **ReactiveX** is a library that implements Observer and Iterator patterns and allows seamless and easy network transactions, manipulate UI events and API responses, on the Web with RxJS, or on mobile with Rx.NET and RxJava. ReactiveX documentation - http://reactivex.io/
4. **Tesseract Tools** for Android is a set of Android APIs and build files for the Tesseract OCR and Leptonica image processing libraries. Provide optical character recognition functionality for word search. Repository and documentation - https://github.com/rmtheis/tess-two
5. **Transitions everywhere** – backport of Android Transitions API for beautiful UI animations (transitions). Used in welcome screen of the app (How To manual). Repository and documentation - https://github.com/andkulikov/Transitions-Everywhere
6. **Datatables** – a plug-in for the jQuery JavaScript library. It is a highly flexible tool that adds advanced features like pagination, instant search, sorting, and filtering to any HTML table. Is used in administration panel on Olympia server. Home page - https://datatables.net/

Support email

olympia8225@gmail.com. Used to send emails to users when they forget the passwords.

Tools and manuals

Actual repositories used for development of Olympia project:

- Github repository for application code and documentation - https://github.com/Sergio9145/Olympia
- Github repository for server code - https://github.com/Sergio9145/Olympia-server
- Cloud 9 - https://ide.c9.io/opryshko/olympia
- MLab database - https://mlab.com/databases/olympia_db
- LucidChart diagrams - https://www.lucidchart.com/invitations/accept/190068c0-2497-4907-9755-e6f65b9c3576

Useful articles, manuals, guidelines and tutorials:

- Tabs adapter - http://www.gadgetsaint.com/android/create-viewpager-tabs-android/#.WxmxFYpKguV
- Swipe adapter - https://medium.com/@ipaulpro/drag-and-swipe-with-recyclerview-b9456d2b1aaf
- Room database manual - https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0
- Optical character recognition - https://solidgeargroup.com/ocr-on-android
- Text to speech - https://o7planning.org/en/10503/android-text-to-speech-tutorial
- Launcher icon generator - https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html
- Icon resources - https://material.io/tools/icons

- Color picker - http://www.color-hex.com/
- Dynamic theme picking - http://www.javarticles.com/2015/04/android-set-theme-dynamically.html
- Animations in Android - https://medium.com/@andkulikov/animate-all-the-things-transitions-in-android-914af5477d50