For developing and controlling version of our project we used Gitlab. This site uses Git software. Git is a widely-used [source code management system](https://en.wikipedia.org/wiki/Version_control_system) for [software development](https://en.wikipedia.org/wiki/Software_development). It is a [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed in 2005 by [Linux kernel](https://en.wikipedia.org/wiki/Linux_kernel) developers (including [Linus Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds)) for Linux kernel development.

As with most other distributed version control systems, and unlike most [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) systems, every Git [working directory](https://en.wikipedia.org/wiki/Working_directory) is a full-fledged [repository](https://en.wikipedia.org/wiki/Repository_%28version_control%29) with complete history and full version-tracking capabilities, independent of network access or a central server. Like the Linux kernel, Git is [free software](https://en.wikipedia.org/wiki/Free_software) distributed under the terms of the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License) version 2. As Git is a distributed version control system, it can be used as a server out of the box. Dedicated Git server software helps, amongst other features, to add access control, display the contents of a Git repository via the web, and help managing multiple repositories.

Also for tracking all the issues and bugs we used YouTrack. It is a proprietary, commercial browser-based bug tracker, issue tracking system and project management software developed by JetBrains. It focuses on query-based issue search with auto-completion, manipulating issues in batches, customizing the set of issue attributes, and creating custom workflows. YouTrack is implemented in compliance with the language-oriented programming paradigm and is developed using JetBrains MPS and a set of JetBrains' own domain-specific languages. It uses a built-in JetBrains database — a transactional key-value store. For remote procedure calls, YouTrack supports a RESTful API.

YouTrack RESTful API allows developers to perform various actions programmatically, including:

* Importing existing projects and issues from other bug tracking systems.
* Creating, modifying, getting all the issues attributes.
* Manipulating projects, users, groups and roles.

At first we develop a prototype of our program in Justinmind Prototyper. Justinmind Prototypingtool is an authoring tool for web and mobile app prototypes and high-fidelity website wireframes. It offers capabilities typically found in diagramming tools like drag and drop placement, re-sizing, formatting and export/import of widgets. In addition, it has features for annotating widgets and defining interactions such as linking, animations, conditional linking, calculations, simulating tab controls, show/hide elements and database simulation with real data. There is support for high-fidelity simulation of Rich Internet Applications, and app prototypes can be simulated on the actual devices too, thanks to Justinmind app. Justinmind can generate HTML prototypes that can be displayed in any browser, and Microsoft Word specification documents.

Working with this software is very easy and we developed a basic view of our app and made a basic list of features that we want to see in our app.

We used ORMLite instead of standard SQLite, because it’s much easier to work with a big list of data and operate data as an object. ORMLite is an [open source software](https://en.wikipedia.org/wiki/Open-source_software) framework that provides lightweight [object relational mapping (ORM)](https://en.wikipedia.org/wiki/Object-relational_mapping) between [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29) classes and [SQL](https://en.wikipedia.org/wiki/SQL) [databases](https://en.wikipedia.org/wiki/Database). It supports [JDBC](https://en.wikipedia.org/wiki/JDBC) databases as well as [Android](https://en.wikipedia.org/wiki/Android_%28operating_system%29) [mobile](https://en.wikipedia.org/wiki/Mobile_phone) platform. The software project was started with the goal of providing a simple yet powerful wrapper around the [JDBC](https://en.wikipedia.org/wiki/JDBC) functions without the complexity of other frameworks such as [Hibernate](https://en.wikipedia.org/wiki/Hibernate_%28Java%29) and [iBATIS](https://en.wikipedia.org/wiki/IBATIS).

ORMLite is [easy](http://ormlite.com/easy_java_orm.shtml) to use and provides the following features:

* Setup your classes by simply adding Java annotations.
* Powerful abstract Database Access Object (DAO) classes.
* Flexible QueryBuilder to easily construct simple and complex queries.
* Supports MySQL, Postgres, Microsoft SQL Server, H2, Derby, HSQLDB, and Sqlite and can be extended to additional databases relatively easily.
* Provisional support for DB2, Oracle, ODBC, and Netezza. Contact the author if your database type is not supported.
* Handles "compiled" SQL statements for repetitive query tasks.
* Supports "foreign" objects with the class field being the object but an id stored in the database table.
* Basic support for database transactions.
* Auto generates SQL to create and drop database tables.
* Spring configuration support for DOAs and class configurations.
* Support for configuring of tables and fields without annotations.
* [Supports native calls to Android SQLite database APIs](http://ormlite.com/sqlite_java_android_orm.shtml)

OrmLite comes with some classes to help manage connections:

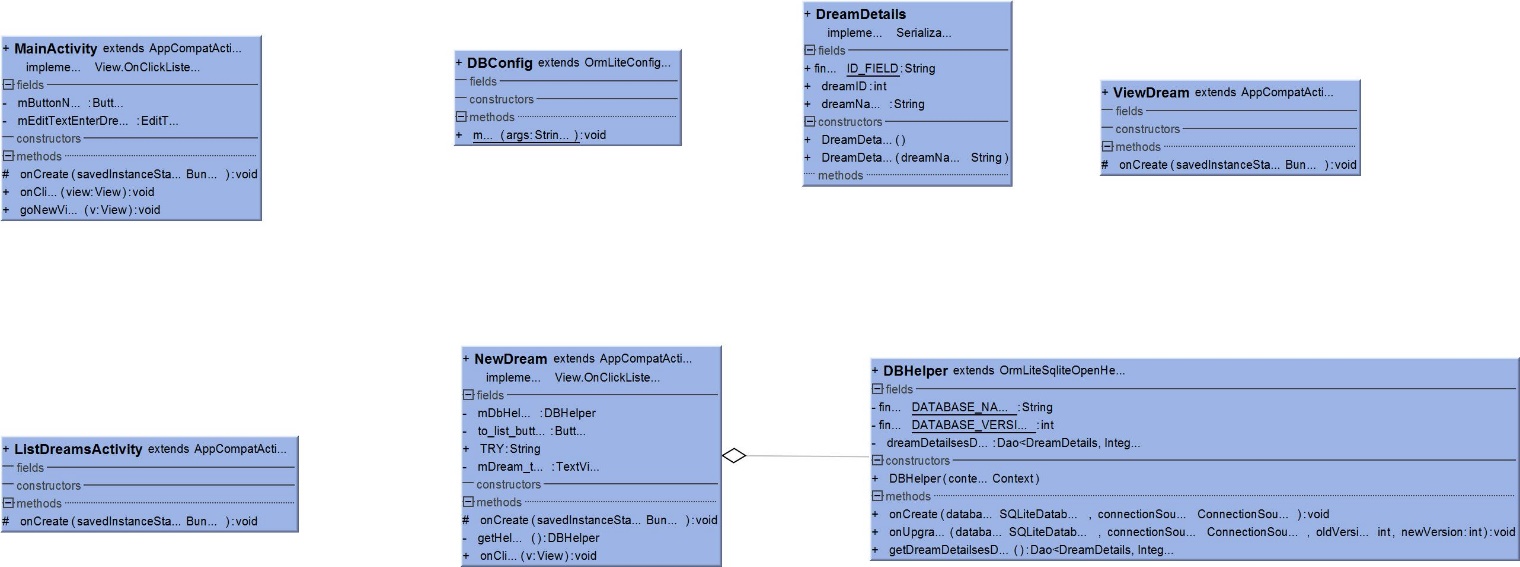
* OrmLiteSqliteOpenHelper (Database Definition). Defines how to connect to the database and create and update the tables. This is where the real connection object lives.
* OpenHelperManager (Connection Management). Keeps count of the number of times the database (connection) has been requested and released. When the count goes to zero it releases the database resources and when it goes back up it recreates them.
* DaoManager (Singleton Management). Ensures that only one instance of a DAO is created for a given connection. Most apps will only connect to one database, so this effectively implements a singleton pattern. It is necessary because creating DAOs is expensive in Android due to the reflection involved.

Program that we developed is called “Dream”. It has 3 classes for operating with database: DBConfig, DBHelper, DreamDetails. And 4 classes for different activities ListDreamsActivity, MainActivity, NewDream, ViewDream.

In database classes we use Singleton pattern, because our program uses only one thread, so we make only on object for operating with database. The singleton pattern is a [design pattern](https://en.wikipedia.org/wiki/Design_pattern_%28computer_science%29) that restricts the [instantiation](https://en.wikipedia.org/wiki/Instantiation_%28computer_science%29) of a class to one [object](https://en.wikipedia.org/wiki/Object_%28computer_science%29). This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects. The term comes from the [mathematical concept of a singleton](https://en.wikipedia.org/wiki/Singleton_%28mathematics%29).

Implementation of a singleton pattern must satisfy the single instance and global access principles. It requires a mechanism to access the singleton class member without creating a class object and a mechanism to persist the value of class members among class objects. The singleton pattern is implemented by creating a [class](https://en.wikipedia.org/wiki/Class_%28computer_science%29) with a method that creates a new instance of the class if one does not exist. If an instance already exists, it simply returns a reference to that object. To make sure that the object cannot be instantiated any other way, the [constructor](https://en.wikipedia.org/wiki/Constructor_%28computer_science%29) is made [private](https://en.wikipedia.org/wiki/Private_member). Note the distinction between a simple static instance of a class and a singleton: although a singleton can be implemented as a static instance, it can also be lazily constructed, requiring no memory or resources until needed.

For making UML class diagram we used SimpleUMLCE plugin for AndroidStudio. This plugin is easy to use and it shows all connections between classes. It also shows all methods and variables of a single class.



Picture 1 – UML Diagram

We also used push notification, as it is necessary for our app to notify user at the time when he wants to and also remind him about his dream. Push notifications let your application notify a user of new messages or events even when the user is not actively using your application. On Android devices, when a device receives a push notification, your application's icon and a message appear in the status bar. When the user taps the notification, they are sent to your application.

We made a connection to Facebook social network, so users can share with their friends their progress of making their dreams come true. This social network is most popular, so almost every user has an account in Facebook. We used Facebook SDK. This SDK allows us to login to Facebook from our app and then post progress in making a dream.