MedBook

Software Architecture Document

Version 1.0

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 27/11/2021 | 1.0 | Complete every section | NPVy, HNMDuc, NKTTNga, NNDu, NPChien |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**Introduction**](#_gjdgxs)4

[**Architectural Goals and Constraints**](#_30j0zll)4

[**Use-Case Model**](#_1fob9te)6

[**Logical View**](#_2et92p0)7

[Component: View](#_tyjcwt) 7

[MainPage view](#_vutl5z24wkxq) 7

[Register view](#_newm223dlk8m) 7

[Login view](#_i5lg6ymu2ig4) 8

[Account view](#_kzyk0z68t8sw) 9

[Home view](#_p63ja5cz7ggl) 10

[Patient activity view](#_kjgdnyh2bxib) 11

[Booking view](#_cla76v75f2df) 12

[Doctor activity view](#_k2in571tca9) 13

[Component: Controller](#_qyinltcjpm89) 15

[MainPageController](#_tfs955h3y1az) 16

[AuthController](#_ttofjnwi6by1) 17

[FireStoreController](#_vg2axujk4tn9) 18

[Component: Models](#_tywacjjzjskh) 19

[Component: Database](#_w4fgz3whgim3) 19

[**Deployment**](#_3dy6vkm)20

[**Implementation View**](#_1t3h5sf)21

[Component: Model](#_u9v8n3fafvrr) 21

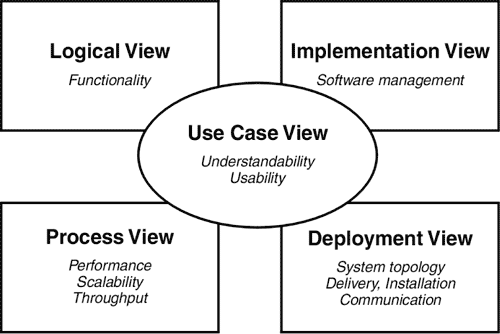
[Component: View](#_nsfbr2ycicq0) 22

[Controller](#_ij6p1lb1zb6q) 23

Software Architecture Document

# Introduction

* 1. **Purpose**
* The purpose of this document is to provide a comprehensive architectural overview of the MedBook Application. It presents a number of different views to depict different aspects of the system.
* In order to depict the software architecture more accurately, the structure of this document is based on the ‘4+1 architectural view model’.



* 1. **Scope**
* The software architecture document applies to each static and dynamic aspect of the system. Since the 4+1 model is used as a reference model, it incorporates many views of the system, thus making the document complete and consistent.
  1. **Definitions, acronyms & abbreviations**
* SAD: Software Architecture Document
* API: Application Programming Interface
* IDE: Integrated Development Environment
* DB: Database
  1. **References**
* Kruchten, Philippe (1995, November). Architectural Blueprints — The “4+1” View Model of Software Architecture. IEEE Software 12 (6), pp. 42-50.
* Object Oriented Software Engineering: A Use Case Driven Approach Ivar Jacobson, Magnus Christerson, Patrik Jonsson, Gunnar Overgaard.
  1. **Overview**

In order to fully understand all the aspects of architecture, the Software Architecture Document contains following sections:

* Section 1: Introduction about the Software Architecture Document.
* Section 2: Describes the architecture objectives and software requirements.
* Section 3: Presented the use-case model of application.
* Section 4: Describes the architecture with components and relationships among them.
* Section 5: Describes how the system is deployed.
* Section 6: Describes the organization for all components of the application (source code, documentations,...)

# Architectural Goals and Constraints

1. **Safety:**

* All doctors in our application are verified by our hospital partners.
* The information about the appointment can be accessed only by users and doctors, we do not share any information with any third parties.
* The connection between users and our server is encrypted by default.
* All sensitive information about doctors and users are encrypted in our database.

1. **Security:**

* When users forget their password, they can reset their password by using the link sent to the registered email.
* If users give us a wrong password 5 times, their account will be disabled in 30 minutes and an email about unauthorized login will be sent to the user's registered email.
* When users log in to a new device we have not known before, there will be an email with a login code sent to users and users need to use that code to log in with that account on a new device.

1. **Performance:**

* The connection bandwidth between our server and client should be greater than 500kbps to make sure the user's experience is the best.

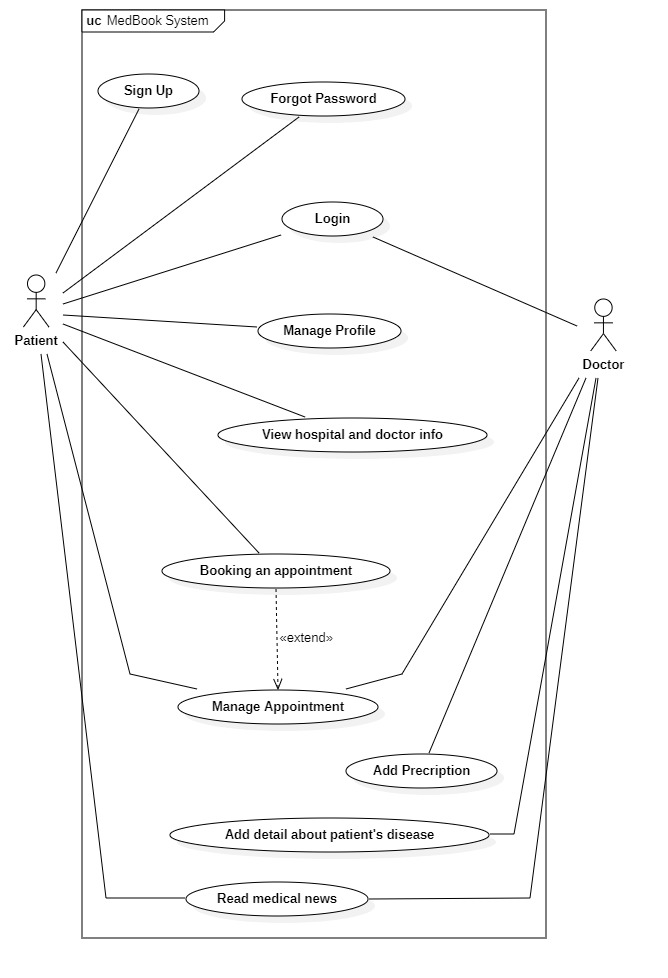
1. **Usability:**

* On the main screen, there are shortcuts to regularly used features.
* The appointment information will be displayed on a separate screen so that users can concentrate on important information.
* There will be a notification before the appointment takes place 2 hours to notify the user.
* All the news and promotions will be displayed on the main screen so that users do not miss any.

1. **Technical platform:**

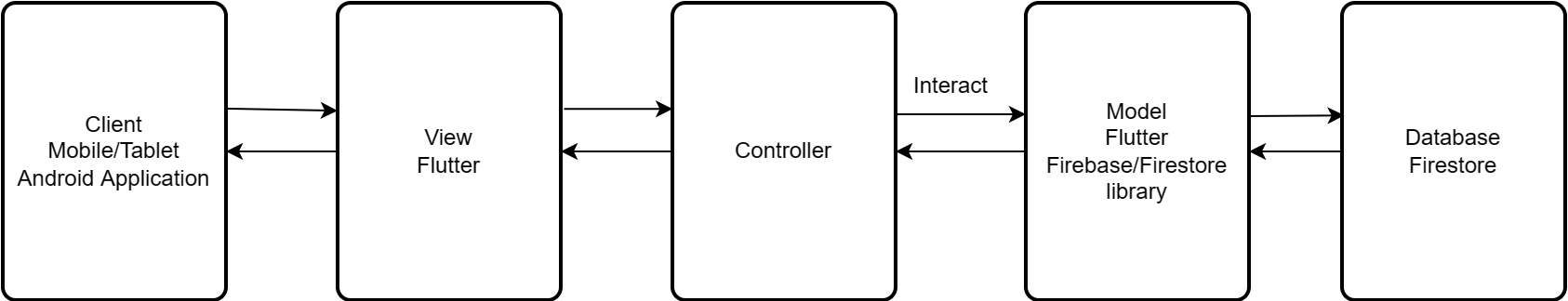
* Operating System: Android
* Version: Android 4.1(minimum requirement) or above.
* Ram: at least 512MB.
* IDE: Android Studio.
* Server database: Firebase.

# Use-Case Model



# Logical View

***This contains 3 big components.***

**

* **Database**: we use Google's service (Firebase, Firestore).
* **Model**: written completely in Dart, handling the interaction between client and server.
* **View**: This component will render all the user interfaces, handling the logic events made by the user (swipe, press,...)
* **Controller**: handling the interaction between view and model. Giving the result to the exact view to show the data. Or get the request from view and send it to the model.

## Component: View

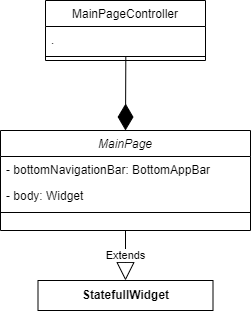
The primary objective of the View Component is to display and update the user interface. This component contains all the functionality that directly interacts with the users and connects with the controllers to handle every user’s event.

### **MainPage view**

* ***Description:***

The MainPage view served as a container that contains the menu bar at the bottom and the body is the current page.

* ***Class diagram:***

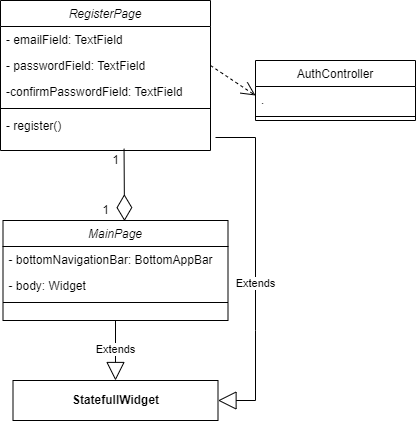
******

### **Register view**

* ***Description:***

The main responsibility of Register view is to provide an interface for users (patients) to register accounts based on their personal information using their preferable methods.

* ***Class diagram:***

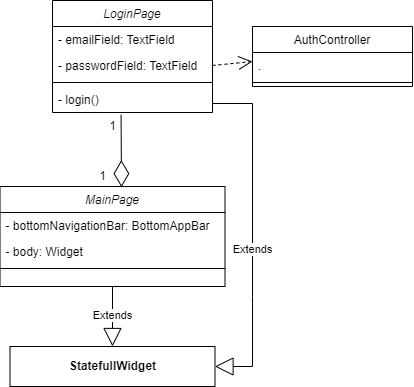
******

### **Login view**

* ***Description:***

The main responsibility of Login view is to provide an interface for users (patients) to log into their accounts based on their personal usernames and passwords using their preferable methods.

* ***Class diagram:***

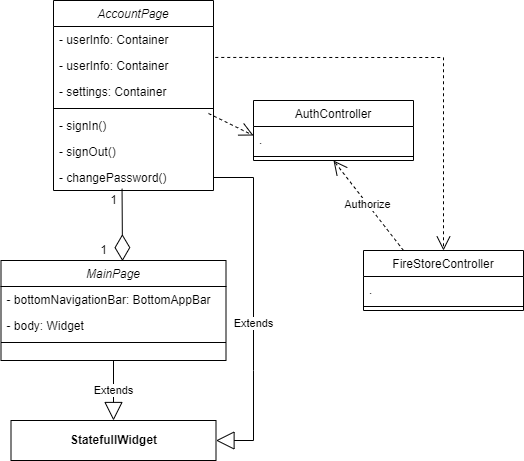
******

### **Account view**

* ***Description:***

The main responsibility of Account view is to provide an interface for users (patients and doctors) to view and edit their accounts based on their personal information.

* ***Class diagram:***

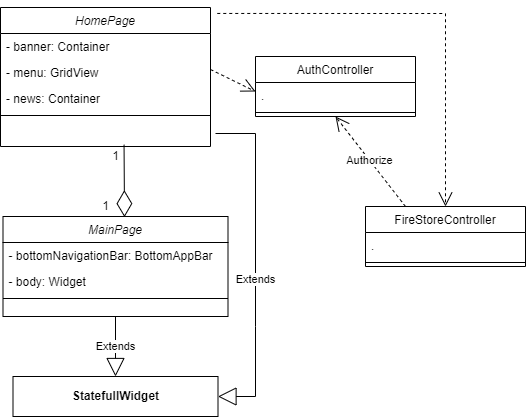
******

### **Home view**

* ***Description:***

The main responsibility of Home view is to provide an interface for users (patients and doctors) to access this application’s functions.

* ***Class diagram:***

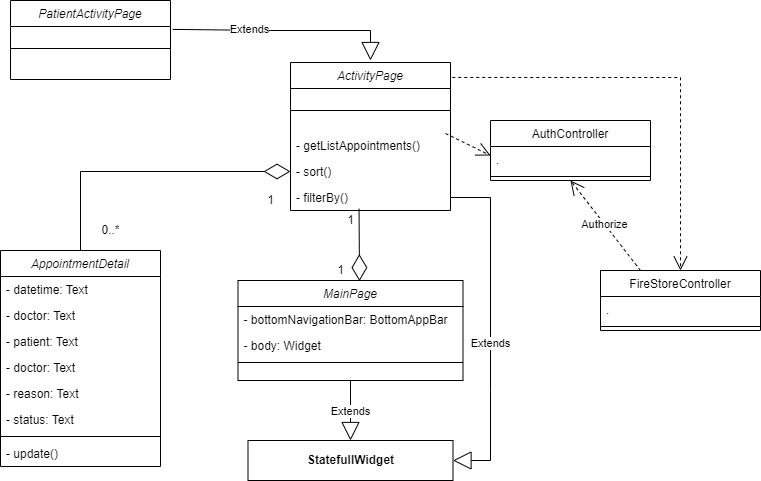
******

### **Patient activity view**

* ***Description:***

The main responsibility of Patient activity view is to provide an interface for users (patients and doctors) to view previous and new appointments.

* ***Class diagram:***

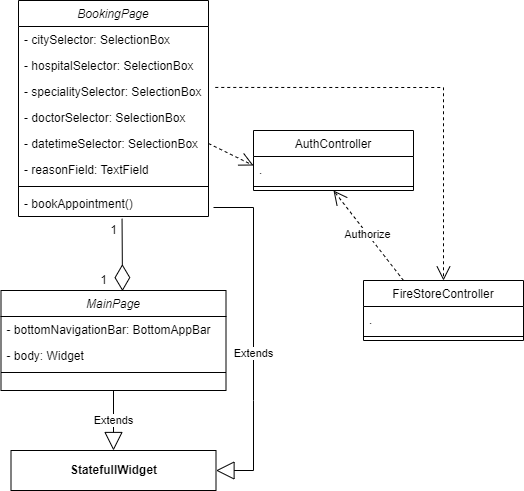
******

### **Booking view**

* ***Description:***

The main responsibility of Booking view is to provide an interface for users (patients) to make appointments based on their expected hospital, doctor, and time.

* ***Class diagram:***

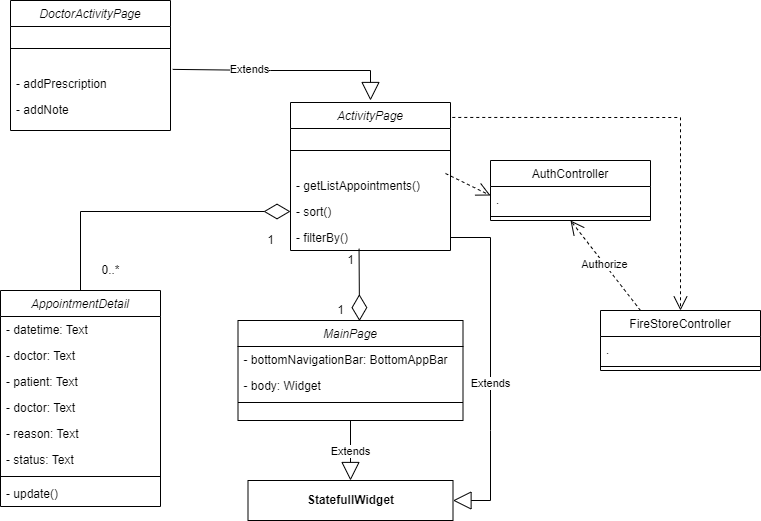
******

### **Doctor activity view**

* ***Description:***

The main responsibility of the Doctor activity view is to display the doctor’s appointment history and upcoming events under a calendar. Doctors can observe appointment detail like patient’s information and meeting time by clicking on that appointment box.

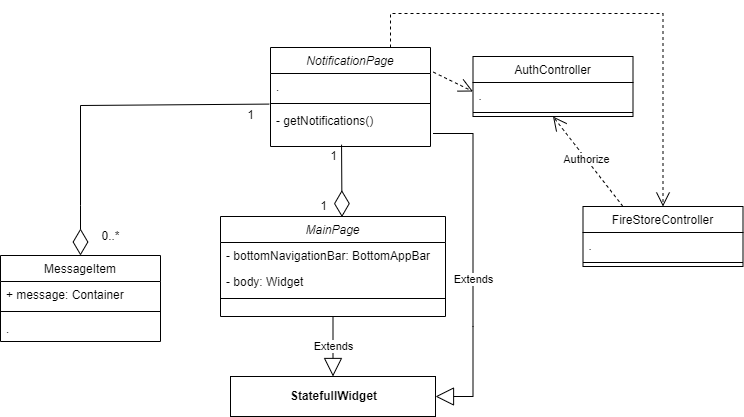
* ***Class diagram:***

******

* + 1. **Notification view**
* ***Description:***

The main responsibility of News view is to display notifications to notify users if they are successful or fail in booking appointments, canceling appointments, or every related event when using this application.

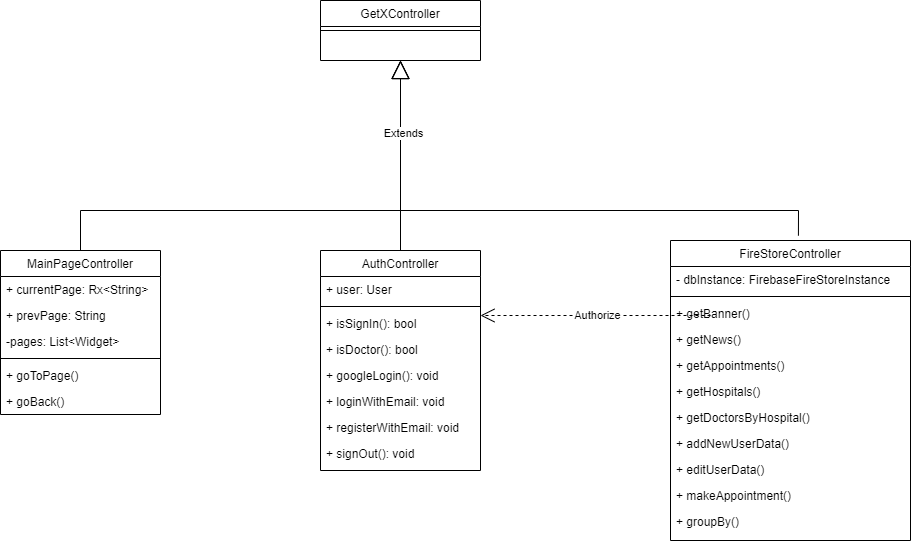
* ***Class diagram:***

******

## Component: Controller

The Controller is the component that connects the model and view, translates every input from the view to a demand for the model to retrieve or update data. To put that in perspective, the View only displays the information, the Controller handles users’ input and interaction by selecting which model types to work with and which view to render data.

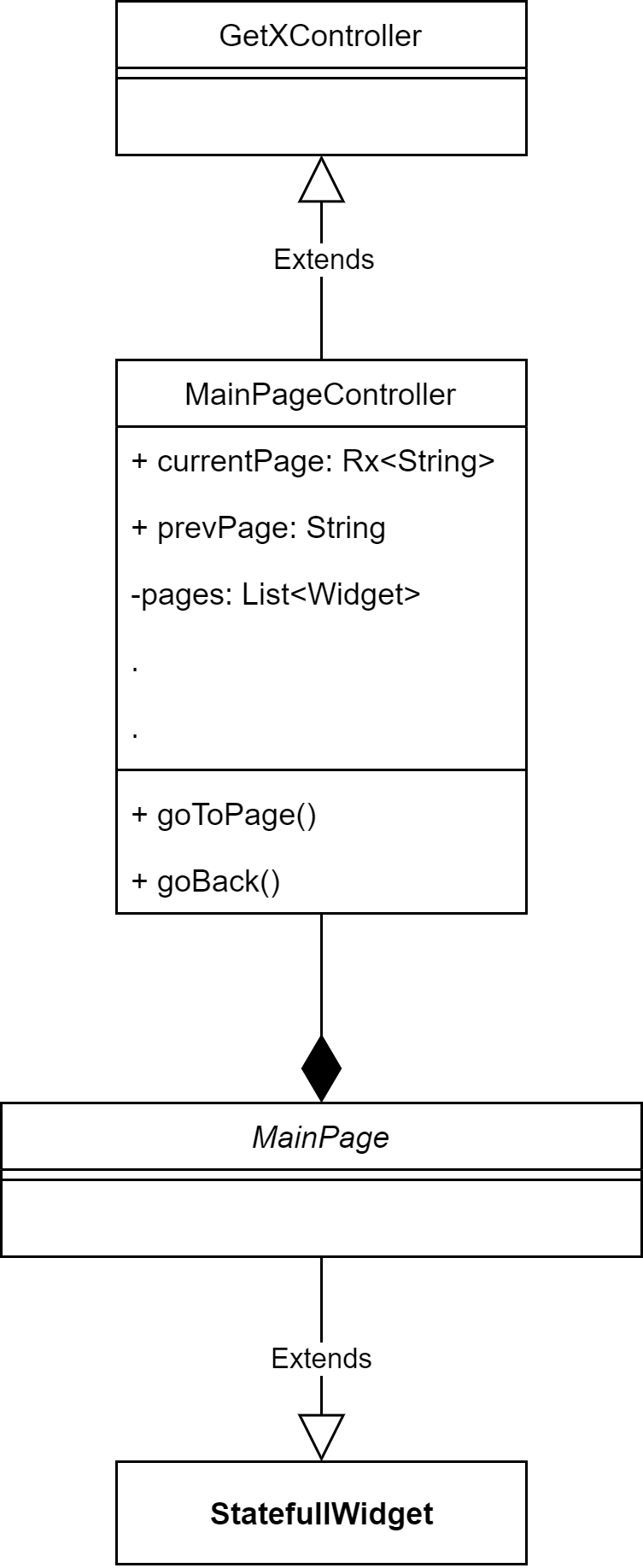
The below image displays three Controllers of this application:



### **MainPageController**

* ***Description:***

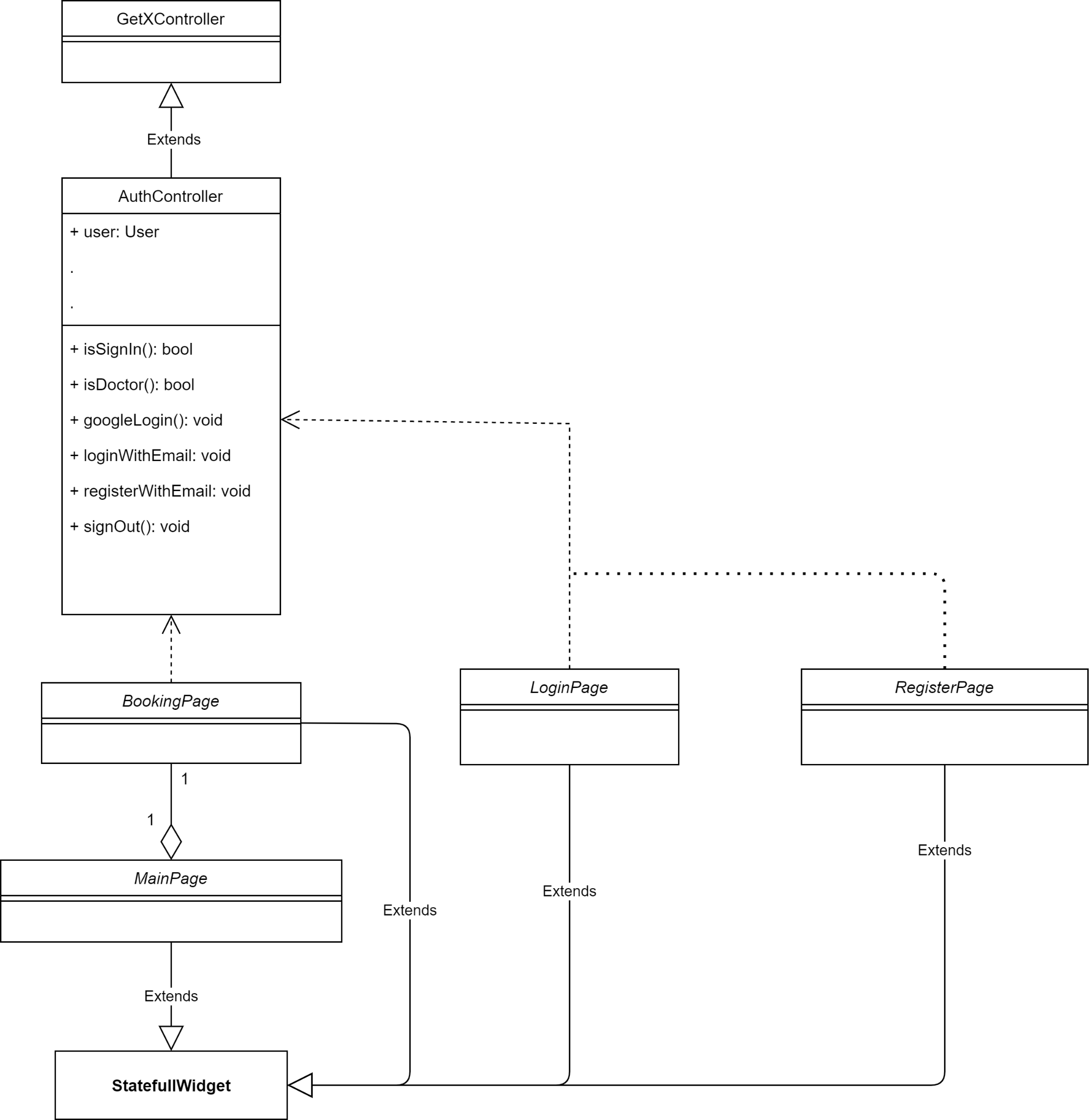
This controller handles interaction between user and application including moving to a selected page, switching between pages or tabs, go-back, and go-forward actions.



### **AuthController**

* ***Description:***

The main responsibility of the AuthController is to control all actions that need to be validated such as sign in, sign out and registration.



### **FireStoreController**

* ***Description:***

The FireStoreController handles action calling to the FireBase database like adding new user, getting appointments list, getting hospital list…



## Component: Models

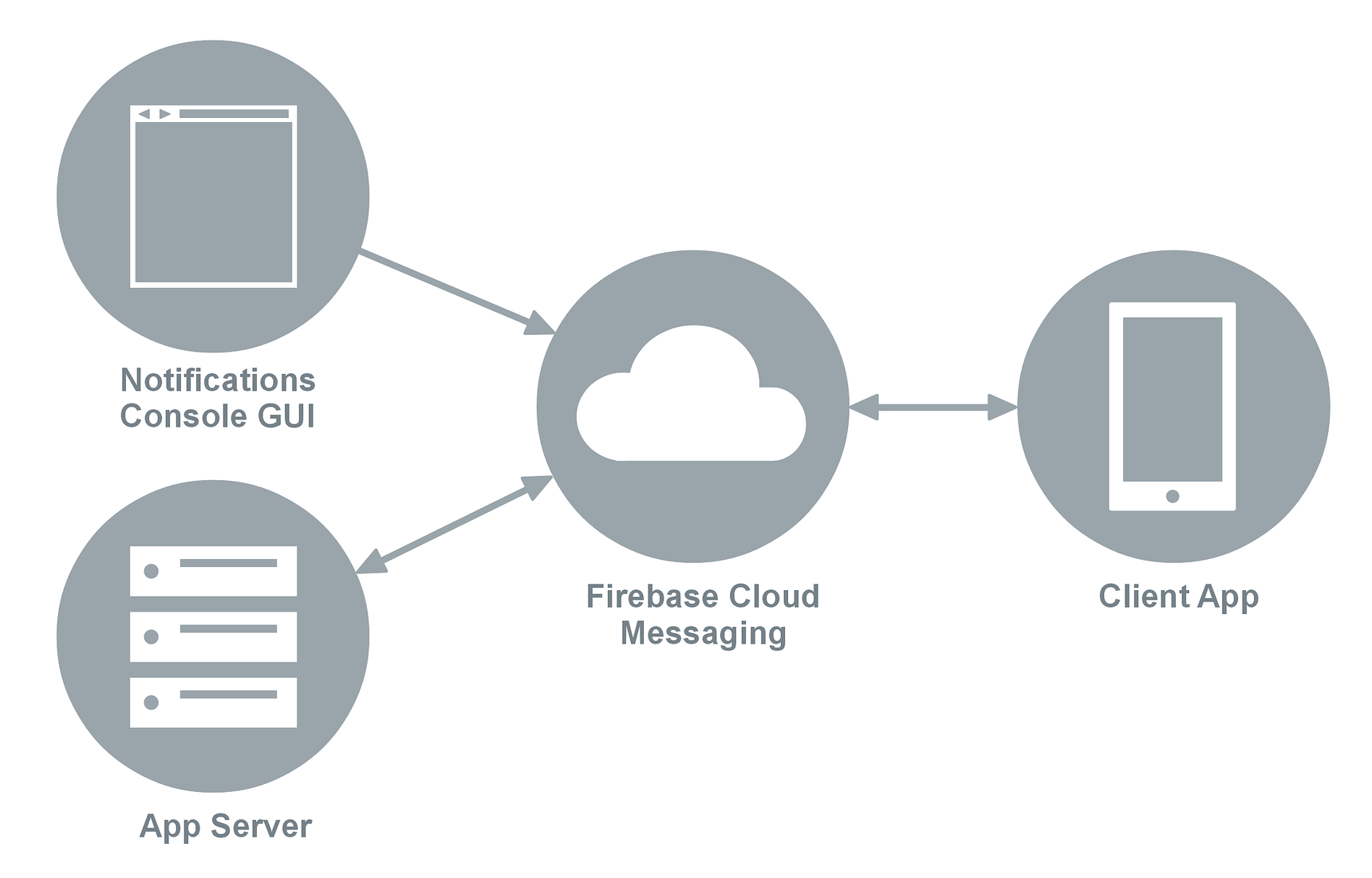
* Patients
* Doctors
* Appointments
* Hospitals

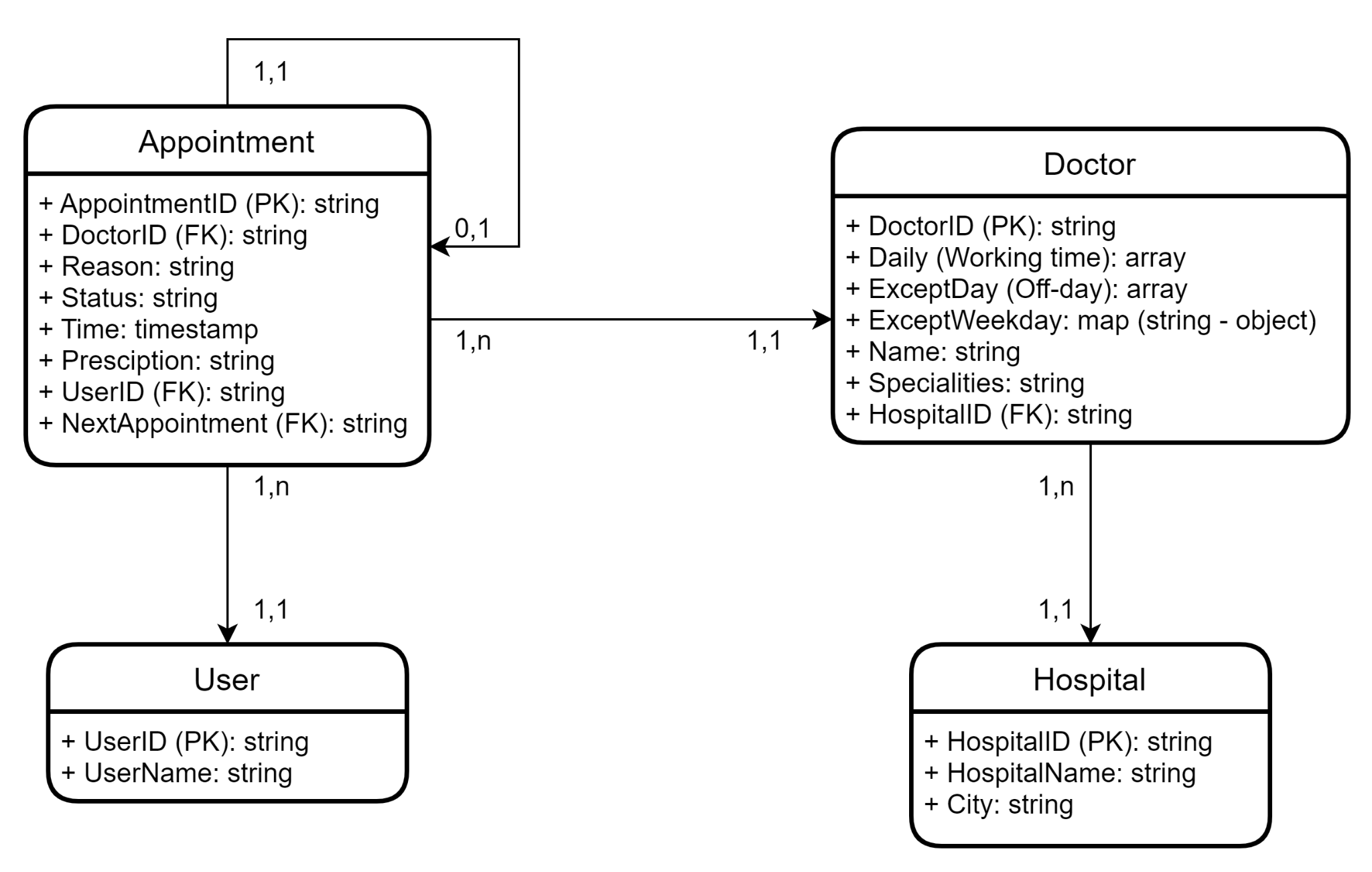
## Component: Database

Using Firebase

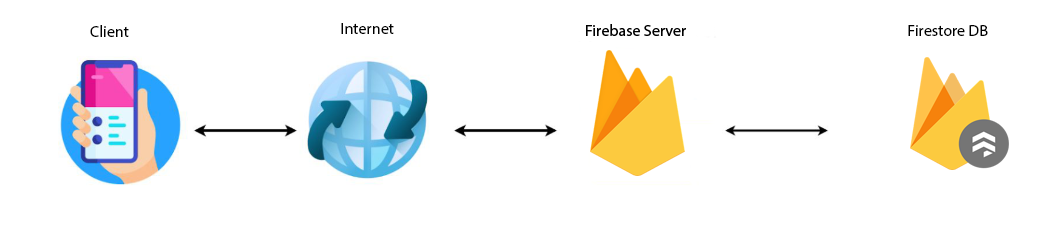
* Responsibilities:

Store all the datas about the application, including user account, appointments dates, doctors’ working days, etc.



**

# Deployment

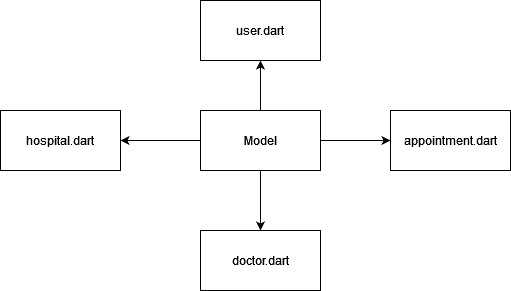
**

Detailed deployment model:

* Our client application interacts directly with Firebase to insert/update/delete information and retrieve data from DB and display it in our application.
* Firestore database will store the account information, appointment information, news and promotion.

# Implementation View

## Component: Model



## Component: View



## Controller

