University of Puerto Rico
Mayagüez Campus

NatAlert

Alejandra Casanova Sepúlveda (802141220)
Christian Santiago (802147568)
Fernando Ortiz(802104952)
Raul Cedres (801141216)
ICOM 4036 Programming Languages
Sección:
Prof. Wilson Rivera

Introduction

There presents a moment where an emergency has occurred in your work area or an area near you. Your responsibility as a manager or supervisor is to notify those employees under you of such emergency, but in the time, it takes you to send an email notifying them of it, your life could be in danger. So, for this purpose an application capable of notifying everyone in a push of a button is needed. If you're an app developer, you understand the knowledge needed to be able to construct such app. And if an extensive knowledge is needed for a graphical user interface, GUI, then more knowledge is needed to be able to tie it to a database, for either safeguarding the notification sent or using the corporation database to use the contact lists for the notification sending function. If you are not an app developer, and your looking for technologies to be able to create an app capable of handling the events mentioned previously then be aware that you may need to research and learn a serious of different programming languages and algorithms to be able to do what NatAlert already does. NatAlert was created to help developers create a GUI app for notification purposes. Purposes as notifying employees of an emergency in their work area, or an area near them. Due to extensive knowledge and time needed to develop such app, NatAlert was born. It uses predefined graphical user interfaces, GUI, templates, this way the developer does not need to invest time in researching how to create or what the GUI needs to be able to function properly. For the developer to maintain a track of the notifications sent, NatAlert implements databases to its features. The database and GUI are already sewn together to further decrease the time invested in research by the developer.

Language Tutorial

Installation:

Python 3.6: For the installation of python 3.6 or higher go to python.org and install the latest update.

Pip: Pip is a tool used for installing python dependencies, if you are installing python 3.6 or higher pip comes already installed in your system.

PyQT5: For installing pyqt5 you just need to go to the terminal and proceed with the following command to install the dependencies to the system.

```
$ sudo pip3 install pyqt5
```

PyreBase: Pyrebase is the library used to communicate with the firebase database used in this language. To be able to install this dependency go to the terminal and proceed with the following command to install the dependency to the system.

```
$ sudo pip3 install pyrebase
```

Ply: For installing the parser for our language go to GitHub and clone the repository for ply.py and yax.py and include them inside your environment.

```
https://github.com/dabeaz/ply
```

NatAlert: For the use of our language go to the NatAlert repository and clone the project to be able to use it on your system.

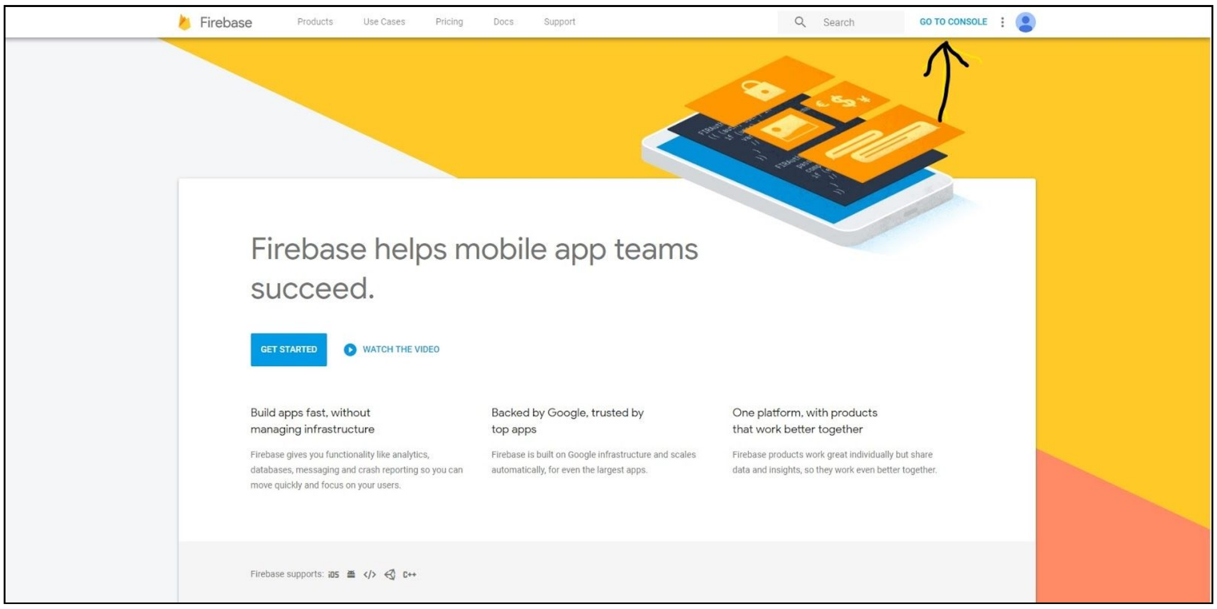
```
https://github.com/CSantiagoBerio/NatAlert
```

Usage:

Writing/Running Code: For the process of writing code open the existing input.txt file available inside the NatAlert repo just downloaded. On that txt file you will find a code example. Proceed to write the necessary code for the purpose of alerting of a specific emergency. After finishing and saving the txt file proceed to run the natparse.py for the system to start reading your recently created code and proceed to make use of you new GUI. For additional changes the user can create multiple txt files but ensure that they are all labeled as input.txt before use inside NatAlert.

Adding a Database:

➔ To add a database to NatAlert, go to Firebase, sign in or sign up if you dont already have an account then click the GO TO CONSOLE button in the upper right corner, right beside the search bar.



Firebase helps mobile app teams succeed.

Build apps fast, without managing infrastructure

Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.

Backed by Google, trusted by top apps

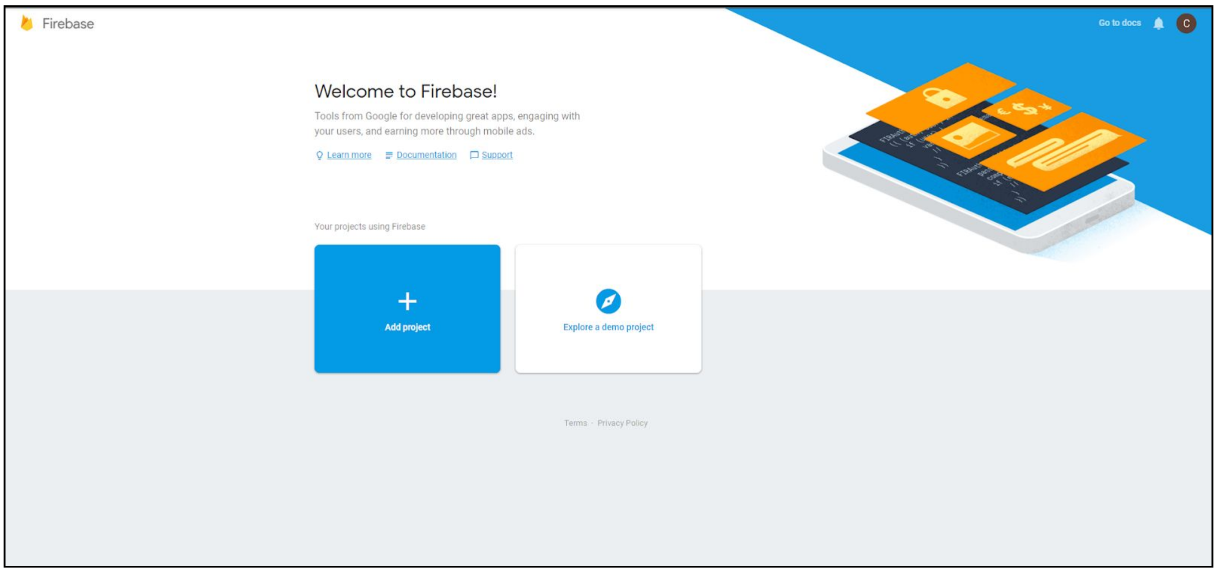
Firebase is built on Google infrastructure and scales automatically, for even the largest apps.

One platform, with products that work better together

Firebase products work great individually but share data and insights, so they work even better together.

Firebase supports: iOS

➔ Add a project, name it whatever you would like.

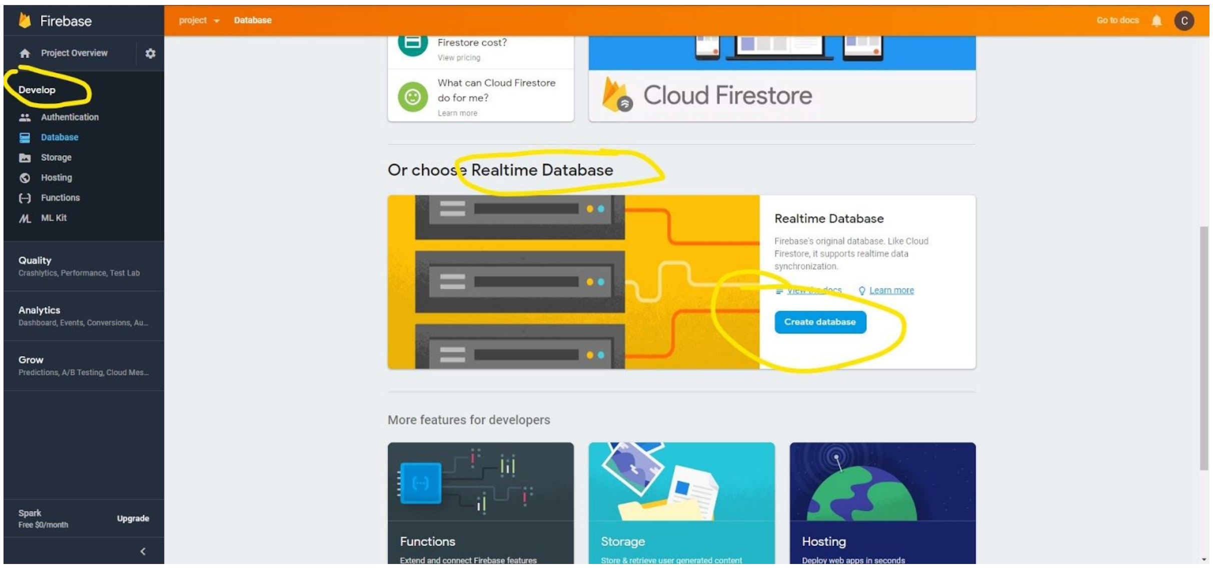


Welcome to Firebase!

Tools from Google for developing great apps, engaging with your users, and earning more through mobile ads.

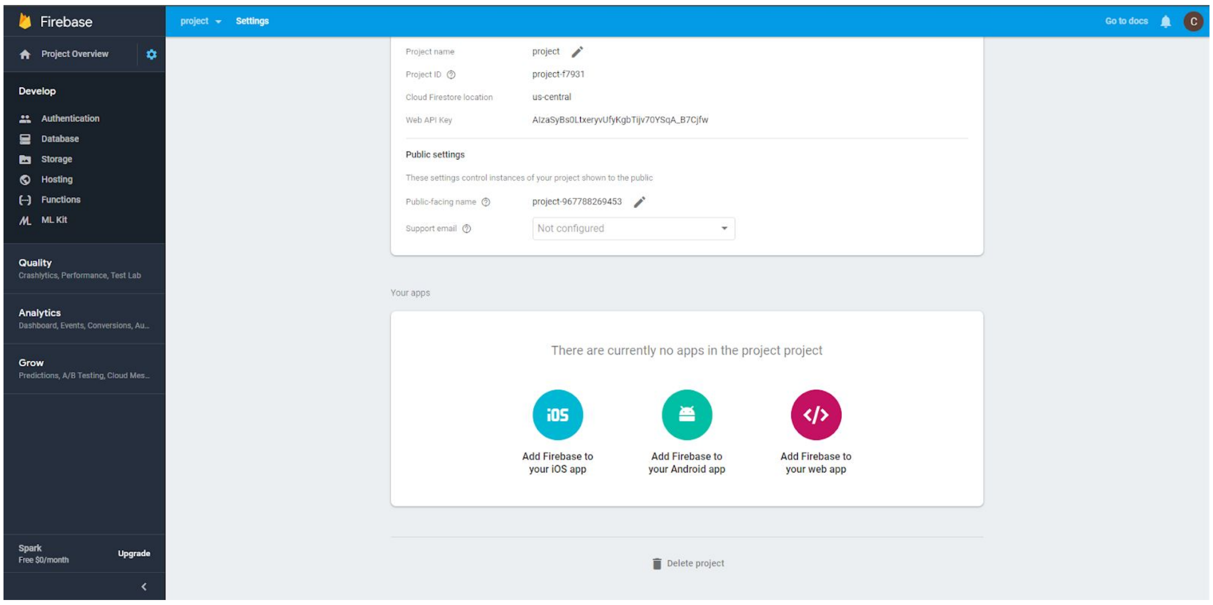
Your projects using Firebase

➔ Go to Develop > Database, create a Realtime Database


Firebase
project
Database
Go to docs
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab
Analytics
Dashboard, Events, Conversions, Au...
Grow
Predictions, A/B Testing, Cloud Mes...
Spark
Free $0/month
Upgrade
Firestore cost?
View pricing
What can Cloud Firestore do for me?
Learn more
Cloud Firestore
Or choose Realtime Database
Realtime Database
View the docs
Learn more
Create database
More features for developers
Functions
Extend and connect Firebase features
Storage
Store & retrieve user generated content
Hosting
Deploy web apps in seconds

Firebase's original database. Like Cloud Firestore, it supports realtime data synchronization.

➔ Go to project settings, click the “Add Firebase to Web App”


Firebase
project
Settings
Go to docs
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test Lab
Analytics
Dashboard, Events, Conversions, Au...
Grow
Predictions, A/B Testing, Cloud Mes...
Spark
Free $0/month
Upgrade
Your apps
Add Firebase to your iOS app
Add Firebase to your Android app
Add Firebase to your web app
Delete project

Project name	project
Project ID	project-f7931
Cloud Firestore location	us-central
Web API Key	AIzaSyBs0LtxeryvUfyKgbTijv70YSqA_B7Cjfw

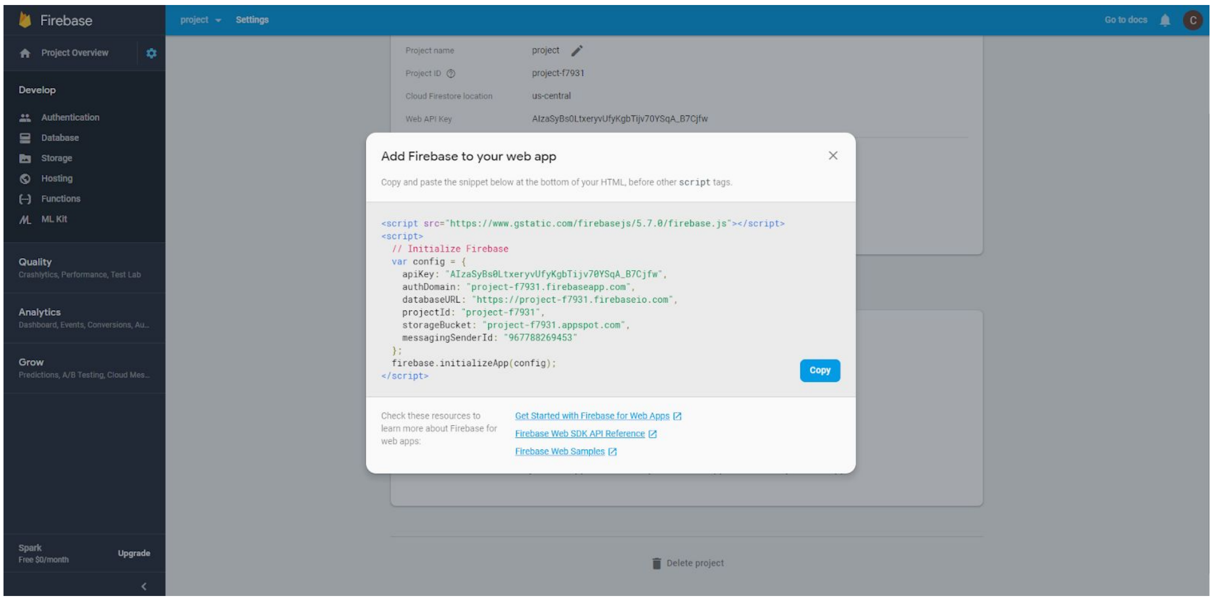
Public settings

These settings control instances of your project shown to the public

Public-facing name	project-967788269453
Support email	Not configured

There are currently no apps in the project project

➔ Copy the variables inside the config variable



Project name project
Project ID project-f7931
Cloud Firestore location us-central
Web API Key AIzaSyBs0LtxeryvUfyKgbTijv70YSqA_B7Cjfw

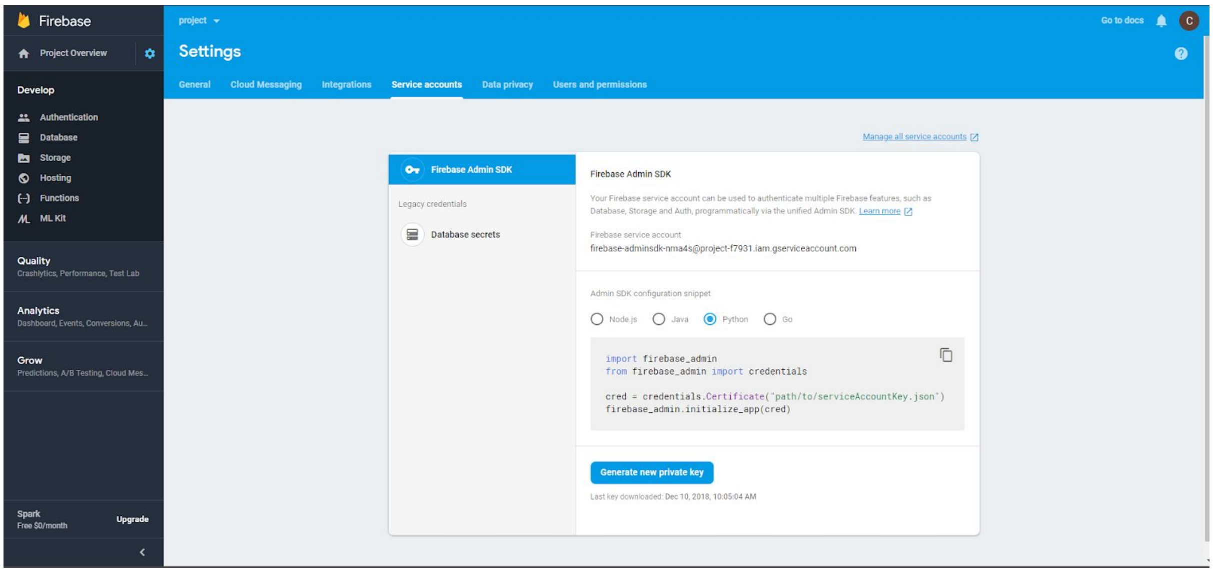
Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other `script` tags.

```
<script src="https://www.gstatic.com/firebasejs/5.7.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyBs0LtxeryvUfyKgbTijv70YSqA_B7Cjfw",
    authDomain: "project-f7931.firebaseapp.com",
    databaseURL: "https://project-f7931.firebaseio.com",
    projectId: "project-f7931",
    storageBucket: "project-f7931.appspot.com",
    messagingSenderId: "967788269453"
  };
  firebase.initializeApp(config);
</script>
```

Check these resources to learn more about Firebase for web apps: Get Started with Firebase for Web Apps; Firebase Web SDK API Reference; Firebase Web Samples

➔ Go to “Service accounts” and click the “Generate new private key” button. This will download a .json file, make sure to know where it is located. Copy this .json file into the json folder in the root folder of the project.



Firebase Admin SDK

Your Firebase service account can be used to authenticate multiple Firebase features, such as Database, Storage and Auth, programmatically via the unified Admin SDK. Learn more

Firebase service account
firebase-adminsdk-nma4s@project-f7931.iam.gserviceaccount.com

Admin SDK configuration snippet

Node.js Java [✓ Python] Go

```
import firebase_admin
from firebase_admin import credentials

cred = credentials.Certificate("path/to/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
```

Last key downloaded: Dec 10, 2018, 10:05:04 AM

➔ Then go to the folder where you cloned NatAlert, go into the UI folder and in the createUI.py file go to line 100, you should see something like in the *Figure A*.

➔ In the fields colored in red, you will input after the colon, the config variables you copied before (Step: *Copy the variables inside the config variable*). In the "*serviceAccount*" field you will input the path to the json file downloaded before, it should start with "json/". My project config settings will look like in the *Figure B*

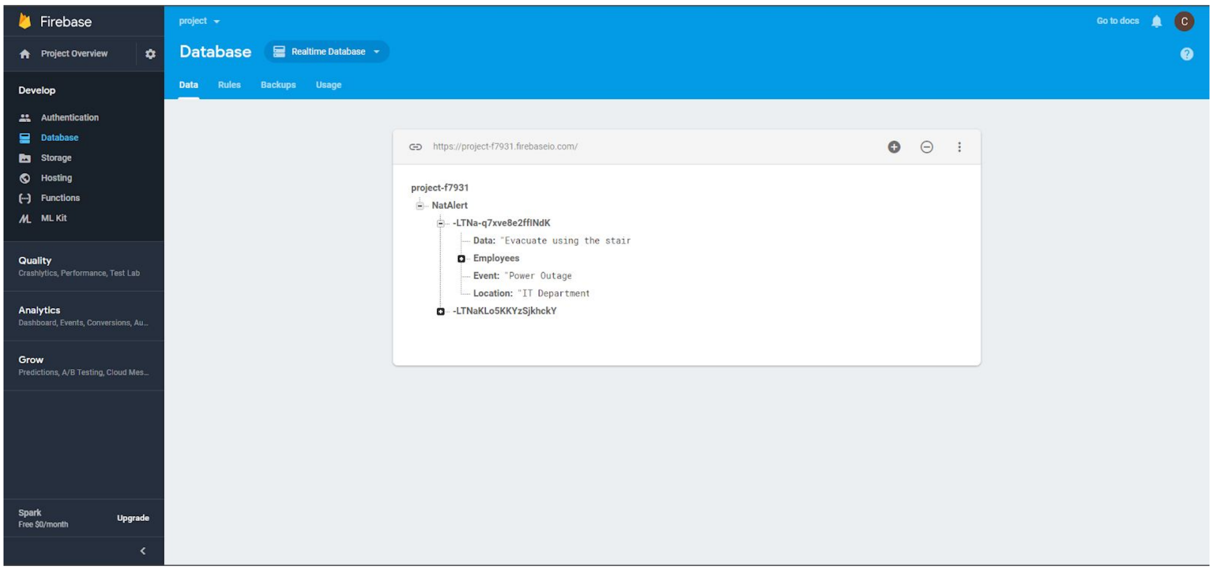
Figure A:

```
def init_DB(self):
  config = {
     'apiKey': "...",
     'authDomain': "...",
     'databaseURL': "...",
     'projectId': "...",
     'storageBucket': "...",
     'messagingSenderId': "...",
     "serviceAccount": "..."
  }
```

Figure B:

```
def init_DB(self):
  config = {
     'apiKey': "AIzaSyBs0LtxeryvUfyKgbTijv70YSqA_B7Cjfw",
     'authDomain': "project-f7931.firebaseapp.com",
     'databaseURL': "https://project-f7931.firebaseio.com",
     'projectId': "project-f7931",
     'storageBucket': "project-f7931.appspot.com",
     'messagingSenderId': "967788269453",
     "serviceAccount": "json/project-f7931-firebase-adminsdk-nma4s-deffd903d3.json"
  }
```

➔ In this step the database should be completely setted up. Just run natparse.py and after selecting the fields in the GUI and clicking the "*Submit*" button, you should see a change in the Database.



```
https://project-f7931.firebaseio.com/

project-f7931
 NatAlert
   -LTNa-q7xve8e2ffINdK
     Data: "Evacuate using the stair
     Employees
     Event: "Power Outage
     Location: "IT Department
   -LTNaKLo5KKYzSjkhckY
```

➔ After running natparse.py and clicking the submit button, you should see an output in the terminal:

```
{ name: " long incoherent string " }
```

➔ This "long incoherent string" is the id of the data you just submitted. Search for this id in the database and you should see the data.

Language Reference Manual

API:

DB.INIT(): Initializes the database for the input of the notification settings in the database

UI.NEW{...}: Sets the settings used by the GUI. It takes a dictionary type parameter.

UI.INIT(): Initializes the GUI. After this function a GUI should pop up, displaying all the settings initialized in UI.NEW{...}

Format

To develop the notification app using NatAlert the format to be used is as follows:

```
DB.INIT();

UI.NEW{
        TITLE: ... ;
        EVENTS: ... ;
        LOCATIONS: ... ;
        SENDTO: ... ;
}

UI.INIT();
```

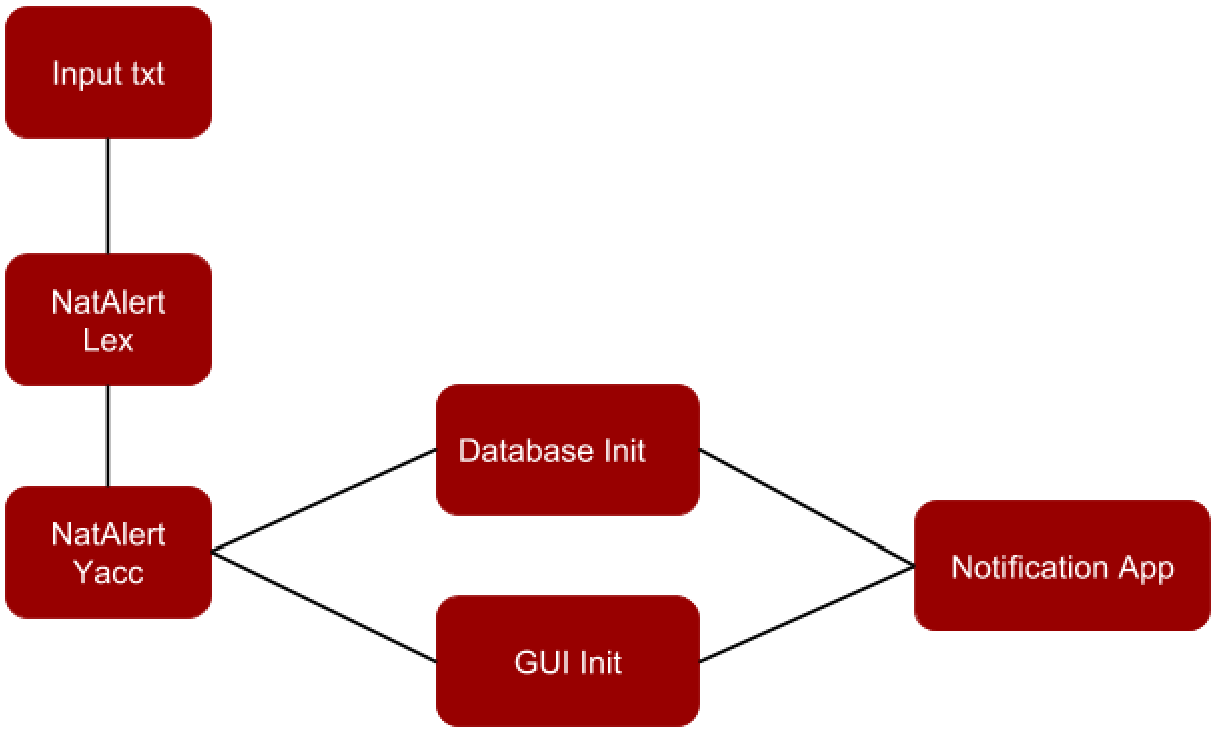
TITLE: The title given to the GUI, it takes a string.

EVENTS: The type of events to be initialized in the GUI, it takes a string or a list of strings.

LOCATIONS: The possible locations where the possible events might happen, it takes a string or a list of strings

SENDTO: The personnel to whom the notification will be sent. It takes a list of strings

Language Development



Input txt:

- ➔ File created with the NatAlert language syntax necessary to develop the Notification App

NatAlert Lex:

- ➔ Uses PLY lex to check if the syntax used is correct

NatAlert Yacc:

- ➔ Uses PLY yacc to parse the text and execute the necessary functions

Database Init & GUI Init:

- ➔ Initializes the Database and GUI interfaces. It makes the connection from the app to the database and initializes the GUI specifications for it to appear and run properly.

Notification App:

- ➔ It is the GUI used for the notification sending. Once it reaches this point the database has been connected to properly and all the settings collected from the input txt file has been parsed and determined to be correctly formed.

Development Environment Set-up:

- ➔ PyCharm IDE
  - ◆ Used for the main development and testing of the programming language.
- ➔ Github
  - ◆ Repository used for the safeguard and storing of the main programming files and folders.
- ➔ PLY
  - ◆ Libraries used for the implementation of a lexer and parser for NatAlert language.

Test Methodology:

The project was divided in four main components: Lexer, Parser, Database and GUI. The test methodology used was a “Code, Test and Debug” methodology. After making a change or implementing new code to a component a test was performed. Each component was tested and debugged separately, after the team confirmed that each component worked correctly as a standalone, the components were integrated to work as one. After every change, a test came along and if necessary a debug followed.

For testing purposes we ran the code and locate where the error appeared. We tested the program using:

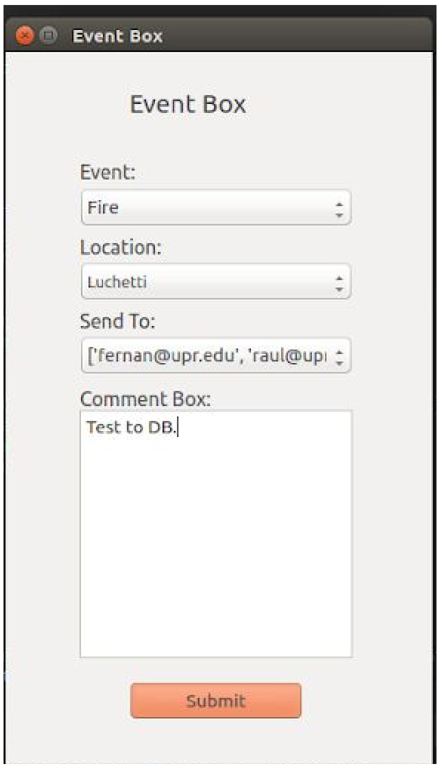
- ➔ PyCharm (for Windows, Mac and Ubuntu)
- ➔ Command Prompt/Terminal

For debug purposes:

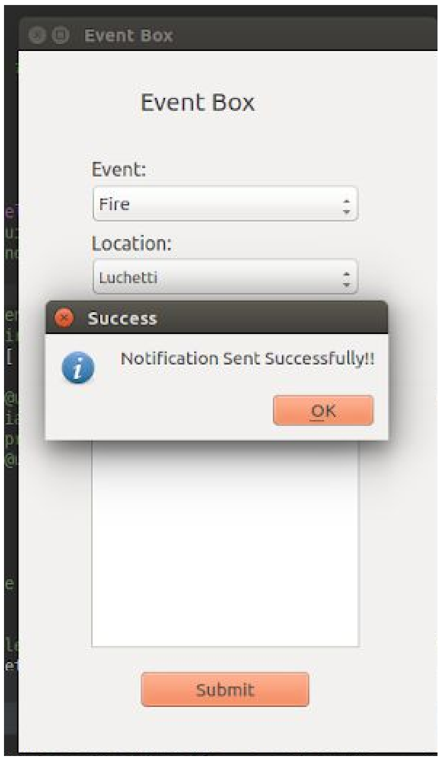
- ➔ PyCharm debug feature

Illustrations of the GUI and Database:

GUI:

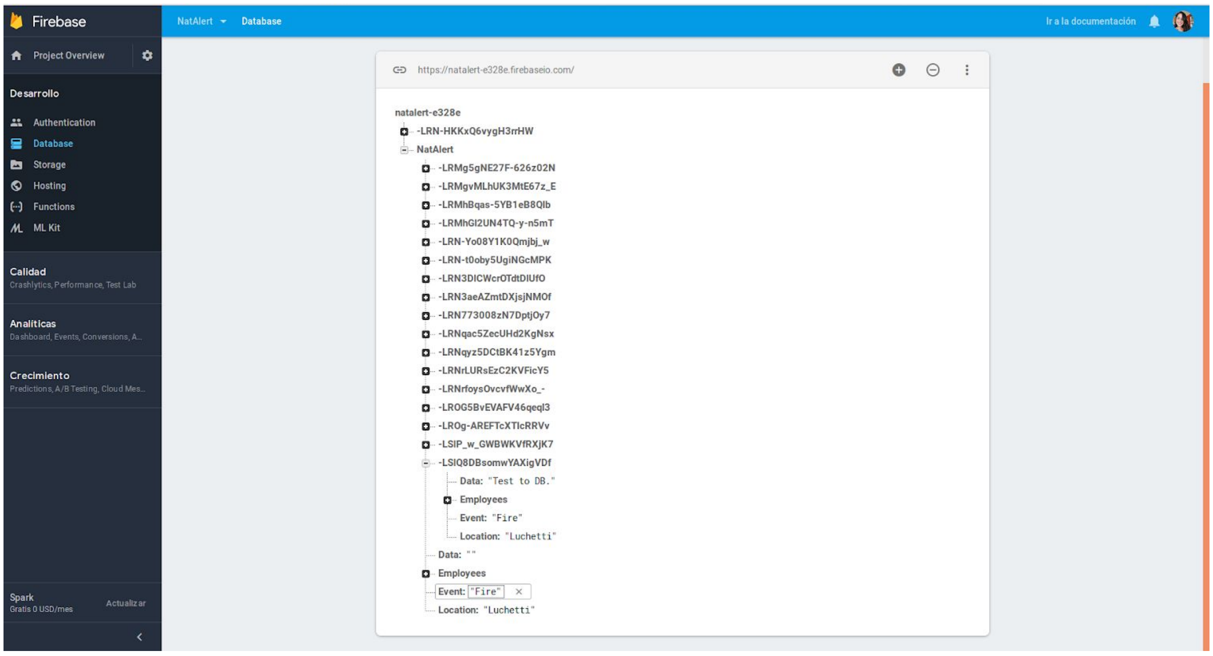


Test to DB.



Notification Sent Successfully!!

Database:



```
https://natalert-e328e.firebaseio.com/

natalert-e328e
  -LRN-HKKxQ6vygH3rrHW
  NatAlert
    -LRMg5gNE27F-626z02N
    -LRMgvMLhUK3MtE67z_E
    -LRMhBqas-5YB1eB8Qlb
    -LRMhGI2UN4TQ-y-n5mT
    -LRN-Yo08Y1K0Qmjbj_w
    -LRN-t0oby5UgiNGcMPK
    -LRN3DICWcrOTdtDIUfO
    -LRN3aeAZmtDXjsjNMOf
    -LRN773008zN7DptjOy7
    -LRNqac5ZecUHd2KgNsx
    -LRNqyz5DCtBK41z5Ygm
    -LRNrLURsEzC2KVFicY5
    -LRNrfoysOvcvfWwXo_-
    -LROG5BvEVAFV46qeql3
    -LROg-AREFTcXTlcRRVv
    -LSIP_w_GWBWKVfRXjK7
    -LSIQ8DBsomwYAXigVDf
      Data: "Test to DB."
      Employees
      Event: "Fire"
      Location: "Luchetti"
    Data: ""
    Employees
    Event: "Fire"
    Location: "Luchetti"
```

Conclusion and Future Work:

NatAlert is a simple to use and has a non-difficult learning curve. It was developed for the purpose of attracting developers and non-developers, as it does not require any programming background. All it needs to function is NatAlert own language syntax, which is easy to learn. Non-developer do not need to know how to develop a GUI or how to connect the GUI components to a specified function or method, all of this is made in the background by NatAlert.

We as developers understand the difficulty of learning various programming languages and various coding algorithms to be able to construct such complex applications. Because of this, we invested our time and knowledge in the development of NatAlert and the simplicity it portrays. NatAlert was made for the construction of notification apps, but it can, for future work, be implemented to develop other kinds of GUI implementing applications. The path of NatAlert is has simply started, a long path lies ahead.

For future work a couple of abilities and functions will be added, such as:

- ➔ More predefined GUI templates.
- ➔ Customization of the templates, such as: color changing, text editing, add/removing components from the GUI.
- ➔ The ability to use the corporation's databases for the emailing list.
- ➔ Notification app notifies personnel through phone notification and not just email.
- ➔ Integration of other databases services.