0.1 API

0.1.1 HTTP

It was decided that an HTTP REST API will be most suitable for this project. Not only that it satisfies all the requirements but it is also extremely popular and widely used nowadays.

HTTP[57] is a text based application layer stateless protocol which uses TCP as an underlying transport protocol. HTTP request contains headers with metadata and an optional body with payload. Stateless protocol means that every request should contain all the necessary data in order to be complete successfully. No information is persisted between the requests, there is no state in the protocol. In the system being developed what is called a server side state[58] is implemented. Server side state has a lot of benefits when compared to the other session management strategies. When using this approach all of the information regarding the state is maintained on the server, the client only has a reference to that data, in our case this reference is called an authorization token and is send with every request as a header. This prevents the client from sending malicious metadata in the session. The session can always be terminated from the backend which will result in the user having to go through the authentication steps again.

The HTTP standard defines several types of requests, called methods, used for a high level distinction between the requests. In the system being developed the following HTTP methods are used:

1. GET - Does not have a body. Consists of headers only. The response contains the payload specified in the request headers. This method is used only for retrieving data. GET request should never modify any data on the server except the caching tags.
2. POST - Can include a request body. This method is used to create a resource. Typical response of a post request is the created entity.
3. PUT - This method is identical to the POST method but is used for modifying a resource.

When transferring data over a network, it has to be encoded in some way. The in-memory platform specific representation needs to be converted to a platform independent format. As it was previously explained HTTP is a text based protocol, which means that the data needs to be represented as strings. For the system being developed a JSON[59] model representation was chosen.

0.1.2 JSON

JavaScript Object Notation (JSON) is lightweight recursively defined string representation of data. Every valid JSON is either:

1. Valid JSON Object
2. Array of valid JSON Objects or Primitives

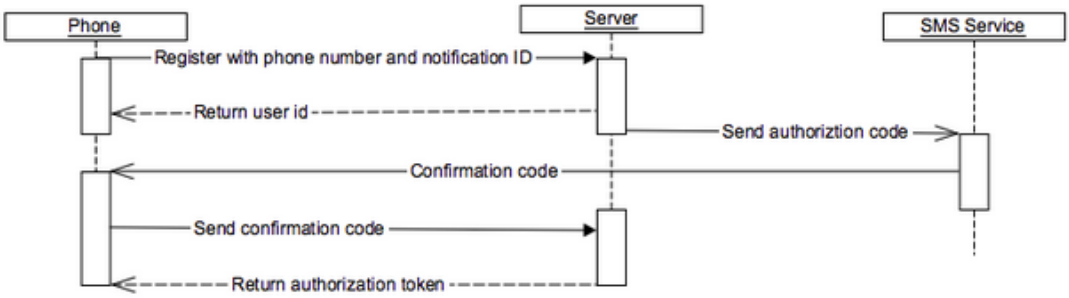
A Primitive is either a String, a Number or a Boolean and every valid JSON object consist of key of valid JSONs or primitives.

0.1.3 Documentation

As the API could be used by third party developers, a proper documentation had to be made. The Swagger Specification [60] is among the most popular specifications for REST APIs. It could auto-generate code for numerous platforms just by following the specification of the API. The swagger documentation consist of JSON/YAML specification of the HTTP Requests. Each request is uniquely identified by its HTTP Method and URL. The documentation describes the data model of the request and response objects using JSON schema[62]. All request and response models along with all the requests are defined in the appendices.

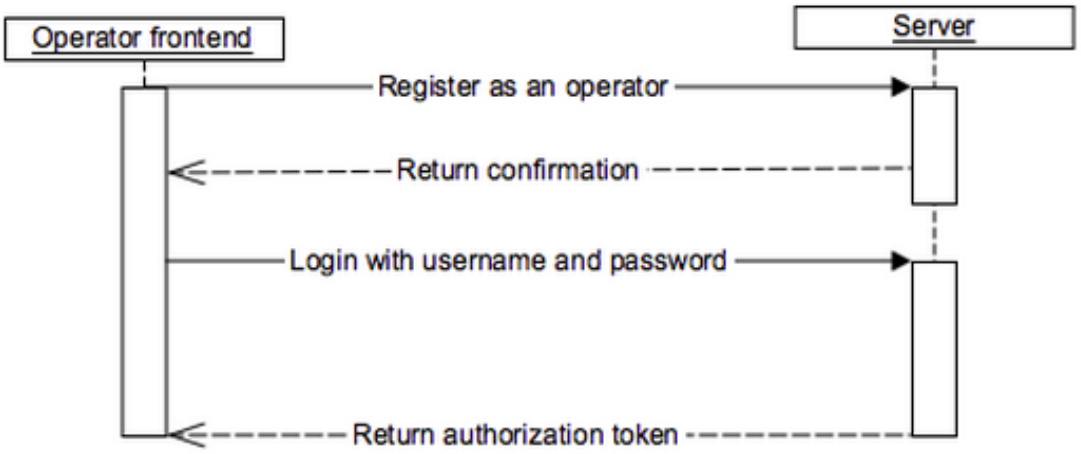
0.1.4 Security

The state of the application is maintained on the server by what is called a server side state. In order to accomplish that, an authorization process needs to be designed.



As shown on the above figure the Phone is authorized by the server using a two-stage authentication process. The mobile device sends its phone number and notification ID for GCM services to the Server. The server then creates an account and associates an authorization code with that account. The phone number and authorization code is then dispatched to the SMS Service to send an SMS to the mobile phone. Once the phone receives the SMS containing the authorization code it sends a confirmation request to the server. The server verifies the confirmation code and if valid - generates and authorization token that is returned to the device. The device uses the authorization token to sign all consecutive requests.

The two-stage authentication process is both easy to implement and secure. The usage of the SMS authorization code ensures that the device being used is real and is associated with the given telephone number. The authorization token is directly mapped to a device and in case of malicious usage of the system can be easily invalidated which will prevent the device from being able to use the system. This is a popular way of authentication when using REST APIs.



A simplified version of the operator register/login routine is shown on the above figure. As with the mobile devices the authorization is again token based. The operator registers as an user of the system providing login credentials such as username and password. Once the registration process is completed, the operator issues a login request to the server. The server then verifies the credentials and if valid - generates an unique authorization token associated with that operator and returns it back. The authorization token represents a login session that identifies an operator. Similarly to the mobile device, the operators application signs all consecutive requests with the authorization token.

The usage of an authorization token provides a way to monitor and limit the number of sessions that the current operator has. If a malicious usage is detected the authorization token can be easily invalidated which will result in preventing the access to the platform for that specific login session.

0.1.5 Requests

All described interactions with the Server in this document, if not explicitly said otherwise, are done using the API. The API contains a /incident/group request that was not described in this document. This request is intended for future use by third party systems with the ability to mark several incidents as identical.