Lohfinder

Lohyna team:
Andrus Oleh
Zhavoronkov Mykyta
Levkovych Roman
Paniush Volodymyr

Agenda

- Introduction | Demo
- Storage
- Resiliency model
- Security model
- Hosted service
- Telemetry
- Monitoring

Intro

Lohfinder

Easily find opportunities to help people and participate in events as volunteers for individual users and seeking for people who will help to create non-commercial events for organizations

Demo | API contract

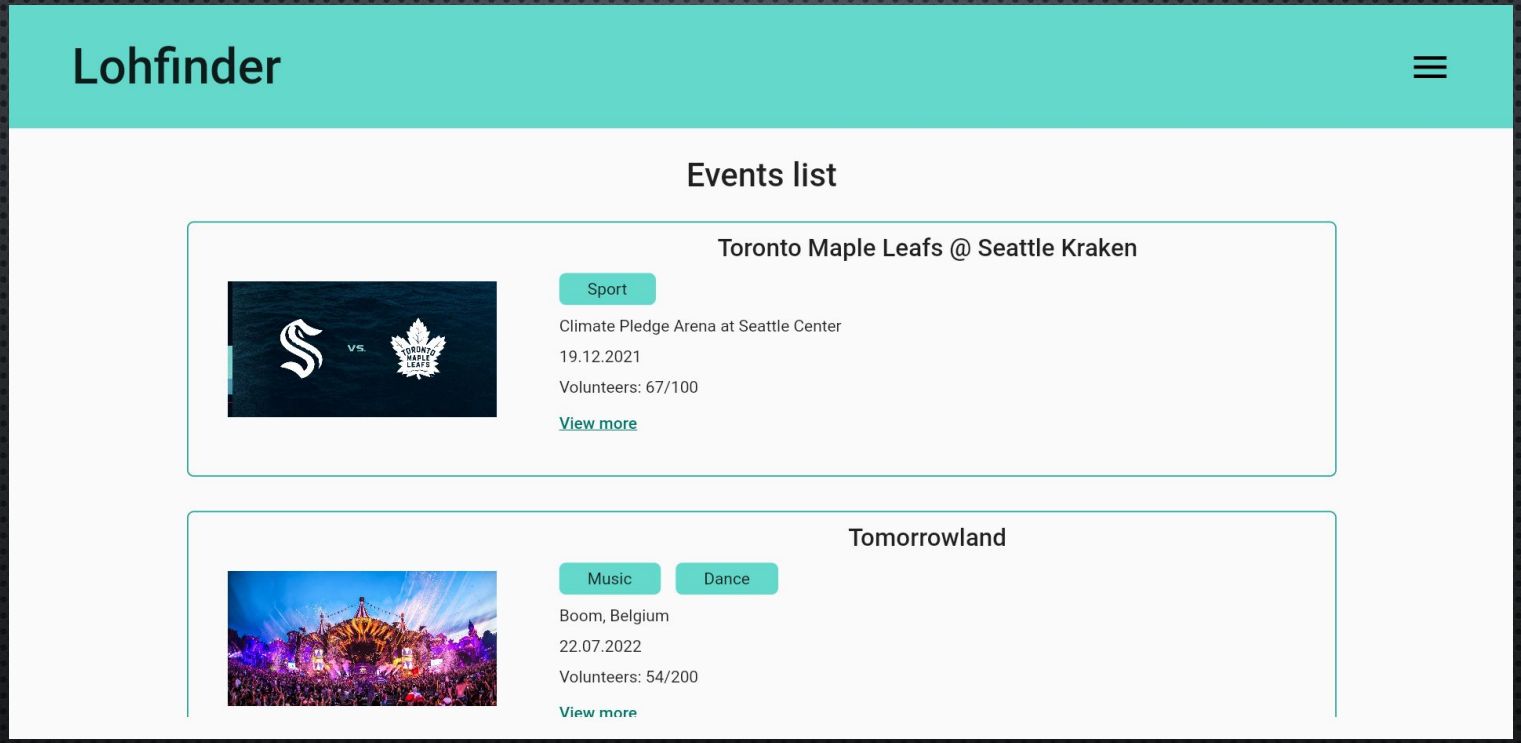
Django REST framework

```
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

{
    "Get all users": "/api/get-all-users",
    "Delete all users": "/api/delete-all-users",
    "Update user": "/api/update-user/<user_id:int>",
    "Save user": "/api/save-user",
    "Get user by id": "/api/get-user-by-id/<user_id:int>",
    "Get user by email": "/api/get-user-by-email/<email:str>",
    "Delete user by id": "/api/delete-user-by-id/<user_id:int>",
    "Delete user by email": "/api/delete-user-by-email/<email:str>",
    "Get all events": "/api/get-all-events",
    "Update an event": "/api/update-event/<event_id:int>",
    "Get event by id": "/api/get-event/<event_id:int>",
    "Delete event": "/api/delete-event/<event_id:int>",
    "Create event": "/api/create-event",
    "Update event application form": "/api/update-event-application/<ea_id:int>",
    "Create event application form": "/api/create-event-application/",
    "Delete event application form": "/api/delete-event-application/<ea_id:int>"
}
```

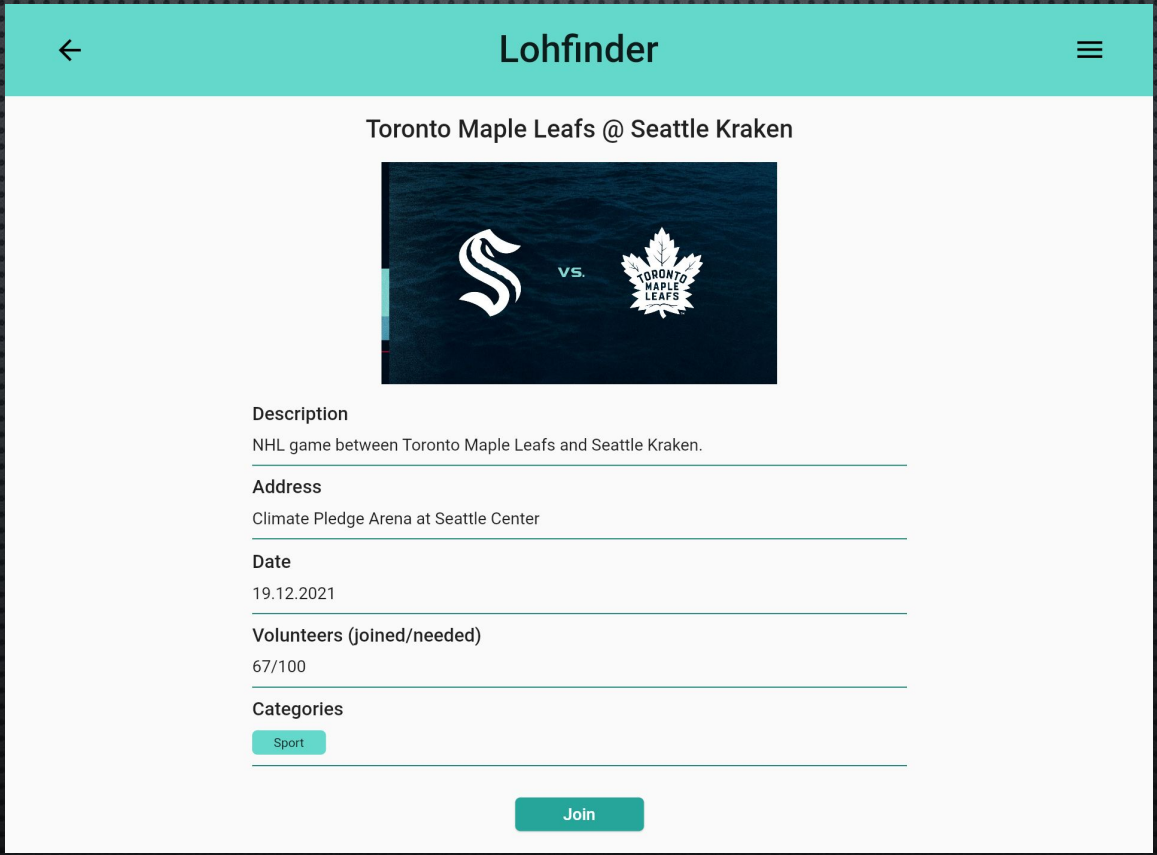
Demo | Dockerized services

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL                                                    docker-compose

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 5/5
 - Container lohfinder_user_service_1          Created                                                    0.0s
 - Container docker_event_application_service  Created                                                    0.0s
 - Container event_service                     Recreated                                                  2.6s
 - Container lohfinder_api_service_1           Created                                                    0.0s
 - Container user_app_service                  Recreated                                                  2.6s
Attaching to docker_event_application_service, event_service, api_service_1, user_service_1, user_app_service
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/users_get-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_get_by_id-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_get_by_email-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_save-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_update-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/users_delete_all-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_delete_by_id-sub
user_service_1                    | Listening messages on projects/lohfinder-app/subscriptions/user_delete_by_email-sub
user_app_service                  |     Finished release [optimized] target(s) in 0.05s
user_app_service                  |      Running `target/release/userappservice`
event_service                     |     Finished release [optimized] target(s) in 0.06s
event_service                     |      Running `target/release/eventservice`
user_app_service                  | Init finished without errors!!!
event_service                     | Init finished without errors!!!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
api_service_1                     | Watching for file changes with StatReloader
api_service_1                     | Performing system checks...
api_service_1                     |
api_service_1                     | Listening messages on projects/lohfinder-app/subscriptions/user_service_result-sub
api_service_1                     | System check identified no issues (0 silenced).
api_service_1                     | December 17, 2021 - 13:10:18
api_service_1                     | Django version 4.0, using settings 'api_service.settings'
api_service_1                     | Starting development server at http://0.0.0.0:8000/
api_service_1                     | Quit the server with CONTROL-C.
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
event_service                     | All statuses successfully sent!
user_app_service                  | All statuses successfully sent!
```

Demo | Web app page sample (Events)



Demo | Web app page sample (Event's info)



Storage | Event Service & Event App Service

SLED : high-performance embedded database with an API that is similar to a BTreeMap<[u8], [u8]>, but with several additional capabilities for assisting creators of stateful systems.

It is fully thread-safe, and all operations are atomic. Multiple Trees with isolated keyspaces are supported with the Db::open_tree method.

ACID transactions involving reads and writes to multiple items are supported with the Tree::transaction method. Transactions may also operate over multiple Trees

Storage | User service application

User service application uses Firebase Realtime database for storing user-profile data. It is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time.

The Realtime Database integrates with Firebase Authentication to provide simple and intuitive authentication for developers. You can use our declarative security model to allow access based on user identity or with pattern matching on your data.

Storage | event application service

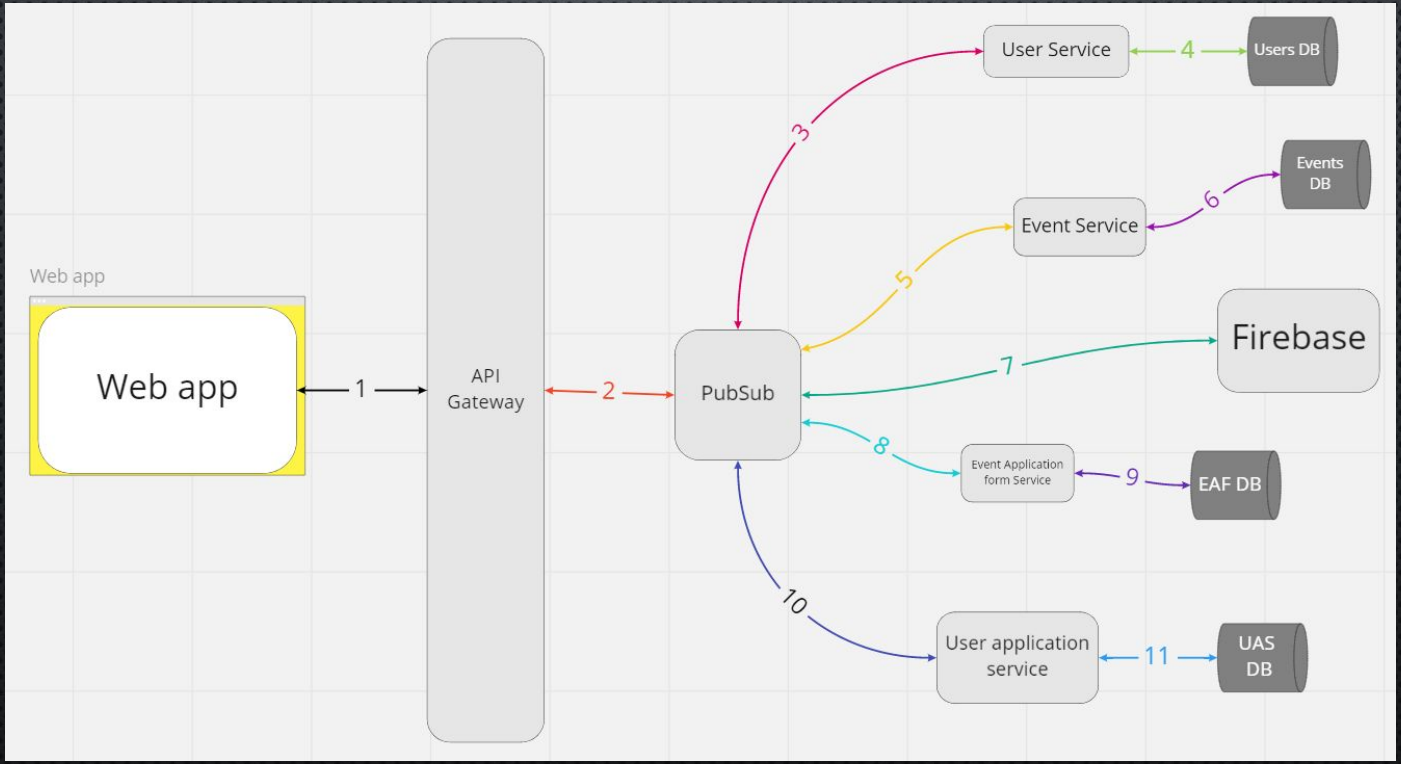
Event application service operates forms that user submits when applying for an event.

So, the service has a lot of semi-structured data to use. That's why it were decided to use NoSQL database.

DB structure:

- Collection per event organizer
- Collection of responses
  - Each response has organizer
  - Each response has the event
  - The response itself

Resiliency model | Component Interaction Diagram



Resiliency model | RMA workbook (pre-work)

ID	Component/Dependency Interaction	Interaction description
1	Web app ←→ API Gateway	Web app sends a REST API request, API Gateway responds to the request.
2	API Gateway ←→ PubSub	API Gateway sends a message to PubSub, PubSub responds to the message.
3	PubSub ←→ UserService	PubSub sends a message to UserService, UserService responds to the message
4	UserService ←→ Users DB	UserService sends a CRUD request to Users DB, Users DB responds to the request.
5	PubSub ←→ EventService	PubSub sends a message to EventService, EventService responds to the message.
6	EventService ←→ Events DB	EventService sends a CRUD request to Events DB, Events DB responds to the request.
7	PubSub ←→ Firebase	PubSub sends a message to Firebase, Firebase responds to the message.
8	PubSub ←→ EventApplicationFormService	PubSub sends a message to EventApplicationFormService, EventApplicationFormService responds to the message.
9	EventApplicationFormService ←→ EAF DB	EventApplicationFormService sends a CRUD request to EAF DB, EAF DB responds to the request.
10	PubSub ←→ UserApplicationService	PubSub sends a message to UserApplicationService, UserApplicationService responds to the message.
11	UserApplicationService ←→ UAS DB	UserApplicationService sends a CRUD request to UAS DB, UAS DB responds to the request.

Resiliency model | RMA workbook (discover). Part 1

ID	Interaction ID	Interaction name	Failure name	Failure description	Response
1	4	UserService ←→ Users DB	Firebase db unreachable	We may lost access to firebase DB due to connectivity issues	Service won't be able to write new changes to DB but will be able to handle read requests with precached info
2	6	EventService ←→ Events DB	No space left on disk	We won't be able to create new events if server won't have enough memory	Service will return errors on any create event request, reading events will work as usual
3	10	PubSub ←→ UserApplicationService	Service unreachable	Pubsub won't be able to reach service if the last will loose connection or will be turned off	All requests which require UserApplicationService won't be handled. When service will be available, all of them will be done
4	2	API Gateway ←→ PubSub	PubSub credentials expire	We can not establish a connection with PubSub if credentials will expire after a certain amount of time	API Gateway (which is a proxy server) can not proceed with the request and send/receive messages from the pool.

Resiliency model | RMA workbook (discover). Part 2

5	1	Web app ←→ API Gateway	SSL certificate error	An error occurs when the browser cannot verify the SSL certificates returned by the server.	When the error happens, the browser blocks the website and warns the user that the website cannot be trusted as shown below. These warnings will negatively impact the user's trust in your website.
6	3	PubSub ←→ UserService	Service unreachable	Pubsub won't be able to reach service if the last will loose connection or will be turned off	All requests which require UserService won't be handled. When service will be available, all of them will be done
7	5	PubSub ←→ EventService	Service unreachable	Pubsub won't be able to reach service if the last will loose connection or will be turned off	All requests which require EventService won't be handled. When service will be available, all of them will be done
8	8	PubSub ←→ EventApplicationFormService	Service unreachable	Pubsub won't be able to reach service if the last will loose connection or will be turned off	All requests which require EventApplicationFormService won't be handled. When service will be available, all of them will be done

Resiliency model | RMA workbook (rate). Part 1

ID	Effects	Portion affected	Detection	Resolution	Likelihood	Rate
3	Major impairment of core functionality or data loss	More than 50%	Between 5 min and 15 min	Between 5 min and 45 min	Multiple times a year	96
6	Major impairment of core functionality or data loss	More than 50%	Between 5 min and 15 min	Between 5 min and 45 min	Multiple times a year	96
7	Major impairment of core functionality or data loss	More than 50%	Between 5 min and 15 min	Between 5 min and 45 min	Multiple times a year	96
8	Major impairment of core functionality or data loss	More than 50%	Between 5 min and 15 min	Between 5 min and 45 min	Multiple times a year	96

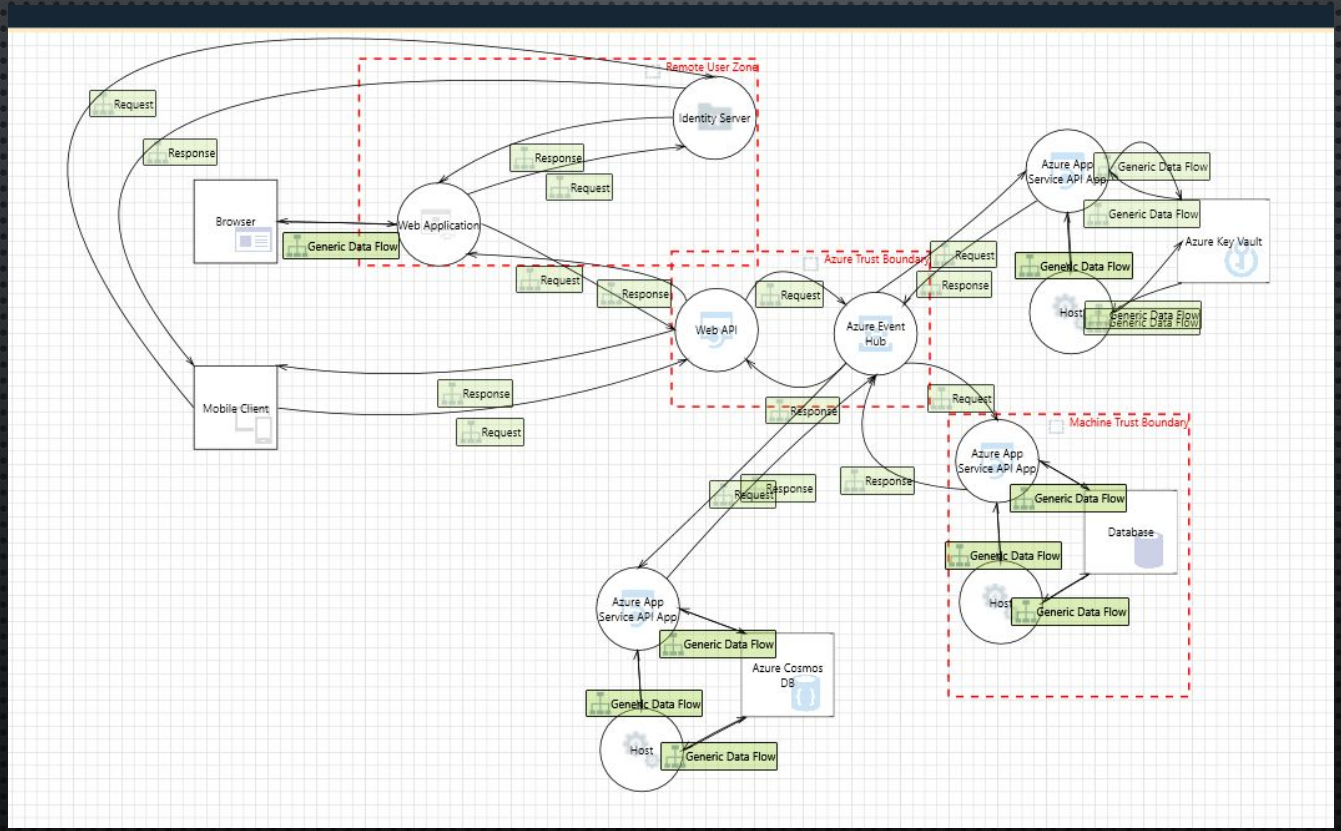
Resiliency model | RMA workbook (rate). Part 2

1	Major impairment of core functionality or data loss	More than 50%	Less than 5 min	Between 5 min and 45 min	Less than once a year	24
2	Some reduction of functionality	Between 2% and 50%	Less than 5 min	More than 45 min	Less than once a year	18
4	Major impairment of core functionality or data loss	More than 50%	Less than 5 min	Less than 5 min	Less than once a year	12
5	Minor impairment	Less than 2%	Less than 5 min	Between 5 min and 45 min	Less than once a year	4

Resiliency model | Act

For improving resilience of our app we wrote unit tests for each service. Also for failures detection we used monitoring tools (see the "Monitoring" chapter).

Security model



Boundaries

1) **Web Application + Identity Server**

   The only endpoint anyone can reach without been properly auth is identity server, where user gets token to reach out Web Api. We are using Firebase Identity Server, which should be secured from malicious attacks, including DOS

2) **Web Api + Event Hub**

   User can reach Web Api only with token he gets from identity server. We assume that if token was successfully validated, command may be passed to Event Hub without security checks. All responsibility on Web Api and separate services

3) **App Service Api + Database + Host machine**

   We’ve deployed each service on dedicated hosts with very limited access to the internet and outer networks. We assume that the only way advisory may negatively affect our service is send command though Event Hub or in any other way spoofing it.

Threats

1) An adversary may gain unauthorized access to Web API due to poor access control checks

Usage of JWT tokens for accessing to Web Api

2) An adversary can gain access to sensitive data by sniffing traffic to Web API

Usage of HTTPS connection to access Web Api

3) An adversary can gain access to sensitive data stored in Web API's config files

The only config files are used in Web Api is Pub/Sub credentials, which lies in Encrypted Filesystem Storage

4) Attacker can deny a malicious act on an API leading to repudiation issues

Full logging implemented on Web Api side, with daily uploading logs to aggregator

Threats

5) An adversary may inject malicious inputs into an API and affect downstream processes

Model verification is done inside Web Api's methods

7) An adversary may jail break into a mobile device and gain elevated privileges

All actions are done on server side. Changing client won't affect security of the application

6) An adversary may spoof Web Application and gain access to Web API

Standard authentication is done between browser and Web Api

8) An adversary can gain access to sensitive information from an API through error messages

We've created separate logging profiles for dev and prod environments

Threats

9) An adversary can gain sensitive data from mobile device

We've encrypted config files to be sure, that user's session won't be stolen or spoofed

10) An adversary can reverse engineer and tamper binaries

Obfuscating of binaries will be done before delivering them to the end users

11) An adversary may gain access to sensitive data from uncleared browser cache

We are not storing sensitive information in browser's cache

12) An adversary can get access to a user's session due to improper logout and timeout

Proper logout is done on Web Api side. Moreover, sessions have finite lifetime

Threats

13) An adversary can spoof the target web application due to insecure TLS certificate configuration

X.509 certificates used to authenticate SSL, TLS, and DTLS connections

14) Attackers can steal user session cookies due to insecure cookie attributes

Secure cookies and HTTPS connection is used

15) An adversary can create a fake website and launch phishing attacks

HTTPS connection is used. All redirects within the application are closed or done safely

16) An adversary can reverse weakly encrypted or hashed content

RSA256 is used for key generation and encryption

Threats

17) An adversary may gain access to sensitive data stored on host machines

Databases will be stored in encrypted file system

19) Sensitive Entity records (containing PII, HBI information) can be inadvertently disclosed to users who should not have access

Splitting users into categories with different access level

18) Secure system configuration information exposed via JScript

Raw JS exceptions are not in use

20) An adversary can gain access to sensitive PII or HBI data in database

Sensitive information is stored in encrypted format, passwords are hashed

Threats

21)An adversary can deny actions on database due to lack of auditing

At least in two services it really could be an issue

22) An adversary may spoof a device by reusing the authentication tokens of one device in another

Only browsers are supported, so it will be only one type of devices

23) An adversary may guess the client id and secrets of registered applications and impersonate them

Asymmetric cryptography is used

24) Spoofing of Destination Data Store SQL Database

This is not handled for now

Threats

25) An adversary may leverage the lack of monitoring systems and trigger anomalous traffic to database

Basic DOS protection will be implemented

26) An adversary may tamper deployed binaries

Binaries are stored on host machines, which are protected by login/passwords

27) PHI at Rest Tampering

No billing available

28) Cross Site Scripting

Web server is protected from XSS attacks

Threats

29) Thread Process Memory Tampered

Raw pointers are not used in any service

30) An adversary can gain unauthorized access to API end points due to unrestricted cross domain requests

Cross domain requests are forbidden

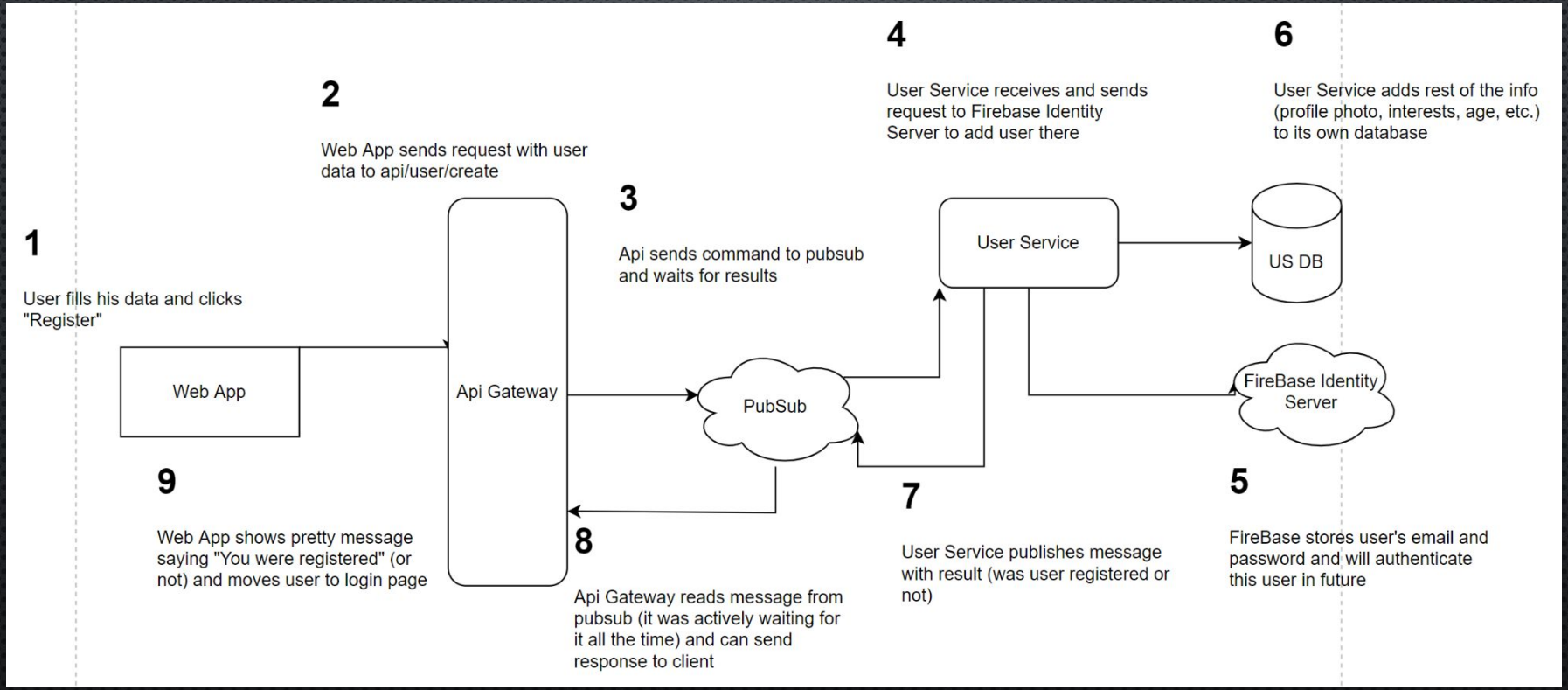
31) A malicious user can deny they made a change to Dynamics CRM

Auditing is enabled in identity server

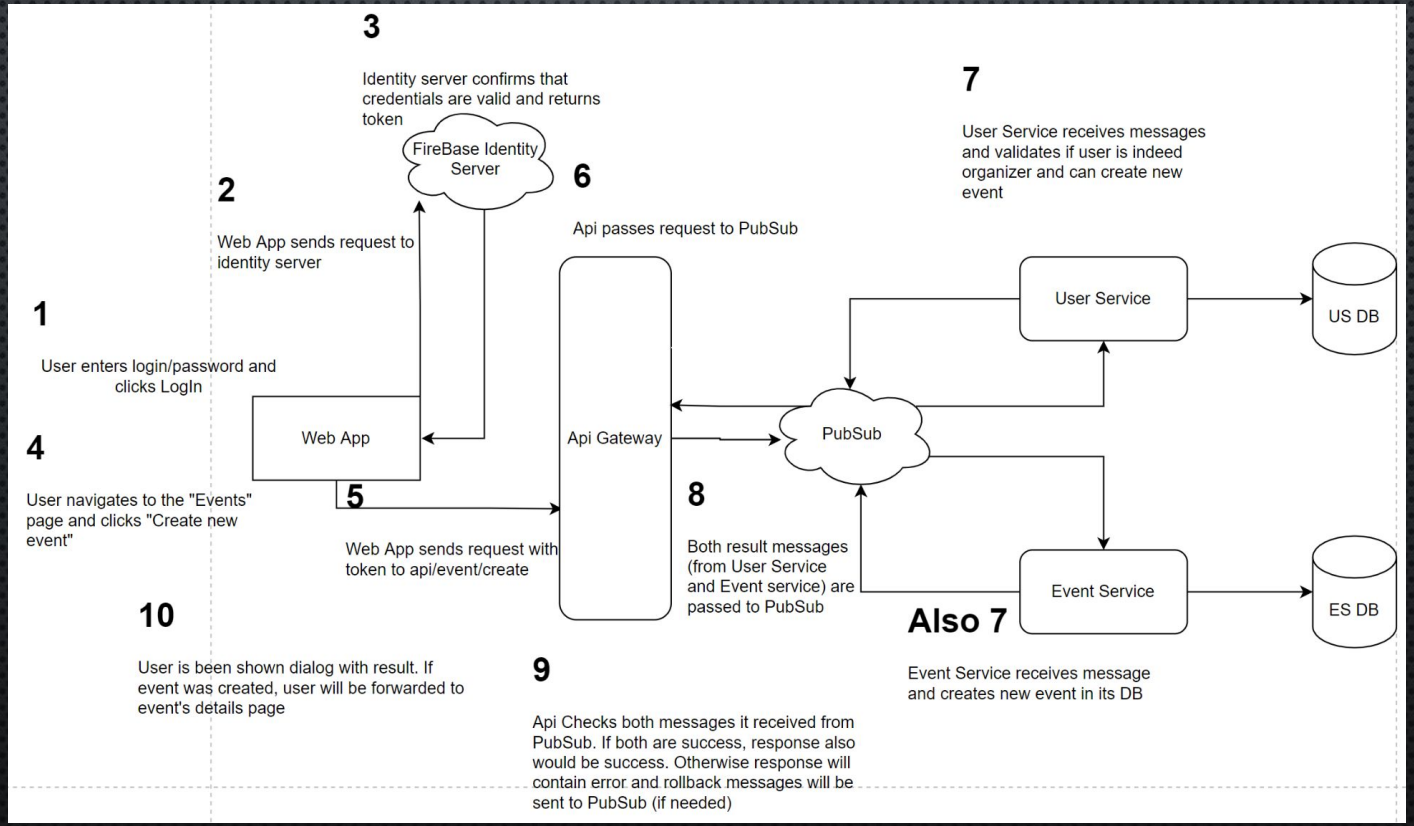
32) An adversary may sniff the data sent from Identity Server

Communication is done through HTTPS

FLows | Registration



FLows | Creating an event

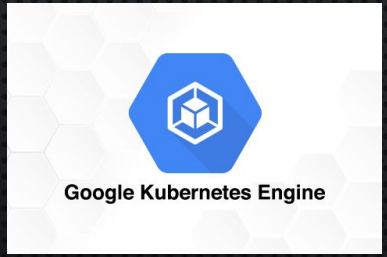


Hosted service | Schema

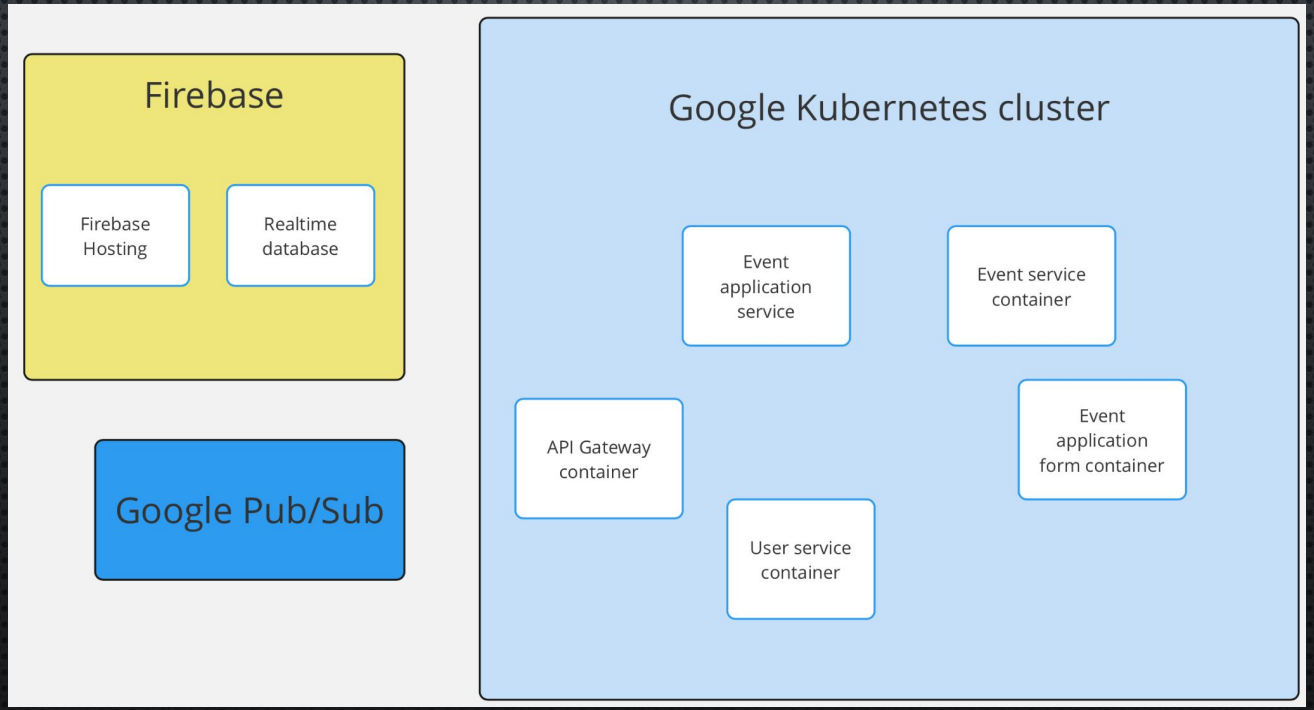
We decided to go with microservices architecture using Google Cloud Platform as IaaS.

GCP services used:

- Pub/sub
- Firebase
- GKE



Hosted service | Schema



Telemetry | Logging

Each service provides logger for tracking user activity and error messages

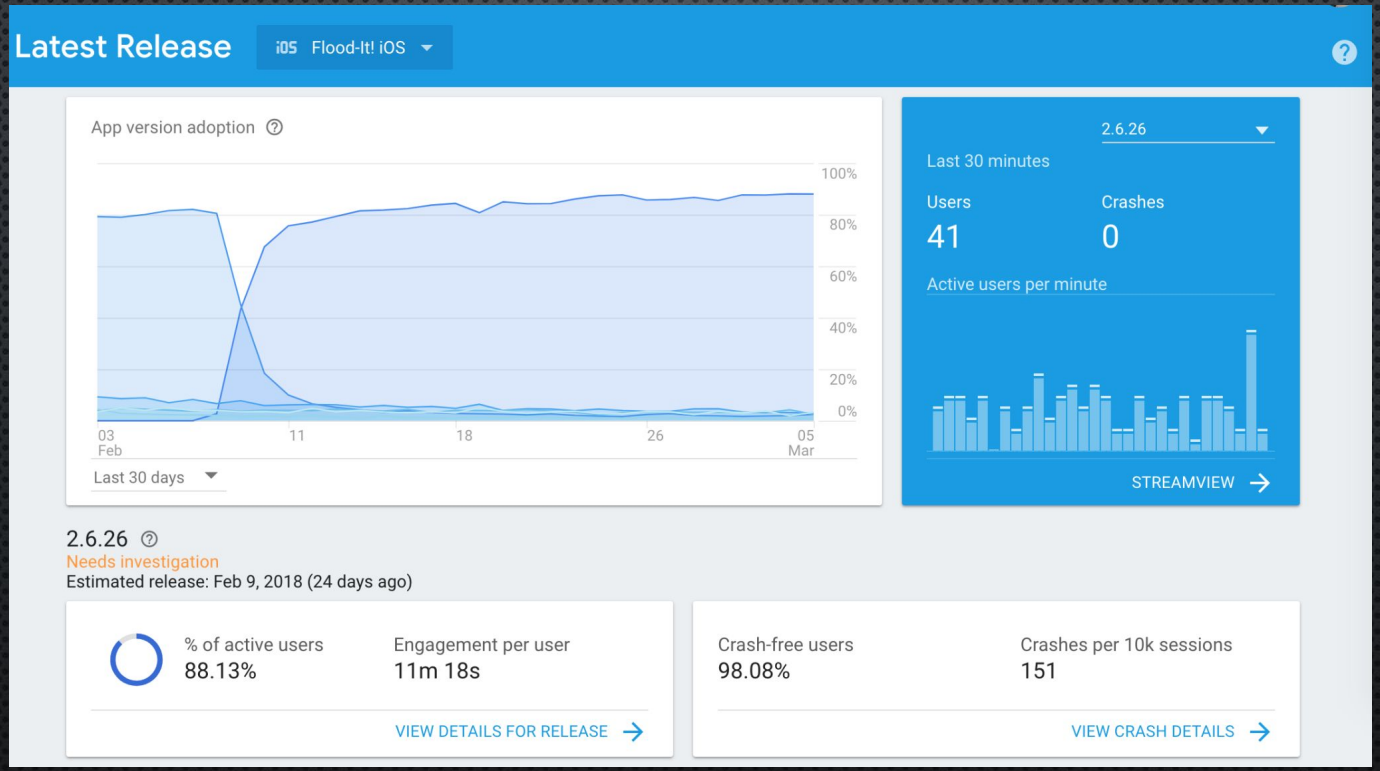
```
api_service_1          | published message: 3681100081352653
user_service_1         | Received message : Message {
user_service_1         |   data: b''
user_service_1         |   ordering_key: ''
user_service_1         |   attributes: {
user_service_1         |     "message_id": "12340597236825159510"
user_service_1         |   }
user_service_1         | }
user_service_1         | published message: 3681099904956467
api_service_1          | Received message : Message {
api_service_1          |   data: b'{"status_code": 200, "result": {"-MohQegcag7JmwaW6...'
api_service_1          |   ordering_key: ''
api_service_1          |   attributes: {
api_service_1          |     "message_id": "12340597236825159510"
api_service_1          |   }
api_service_1          | }
api_service_1          | [16/Dec/2021 17:58:15] "GET /api/get-all-users/ HTTP/1.1" 200 8219
```

Telemetry | Firebase Performance Monitoring

As a dashboard for user activity we will use Firebase Performance Monitoring that can be used to:

1. Automatically measure app startup time, HTTP network requests, and more
2. Gain insight into situations where app performance could be improved
3. Customize monitoring for your app

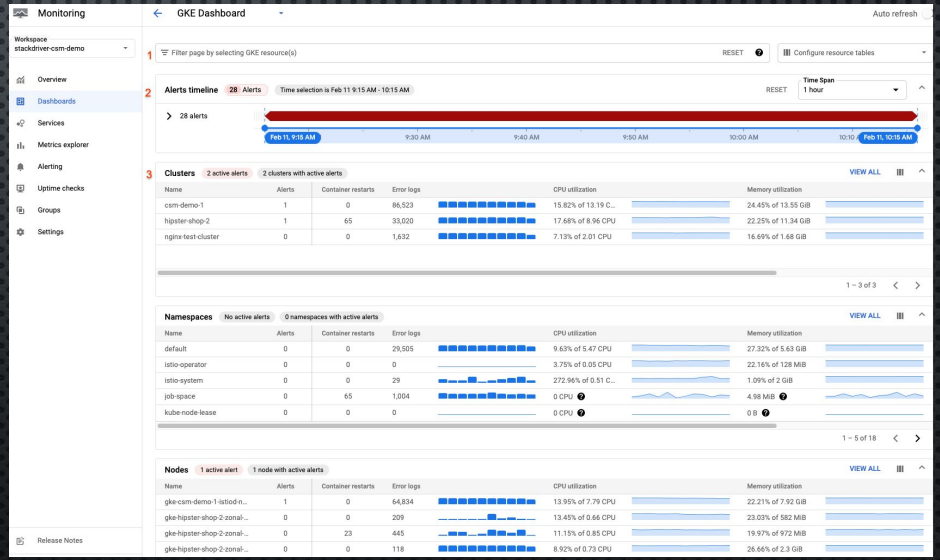
Telemetry | Firebase Performance Monitoring



DASHBOARD EXAMPLE

Monitoring | GKE

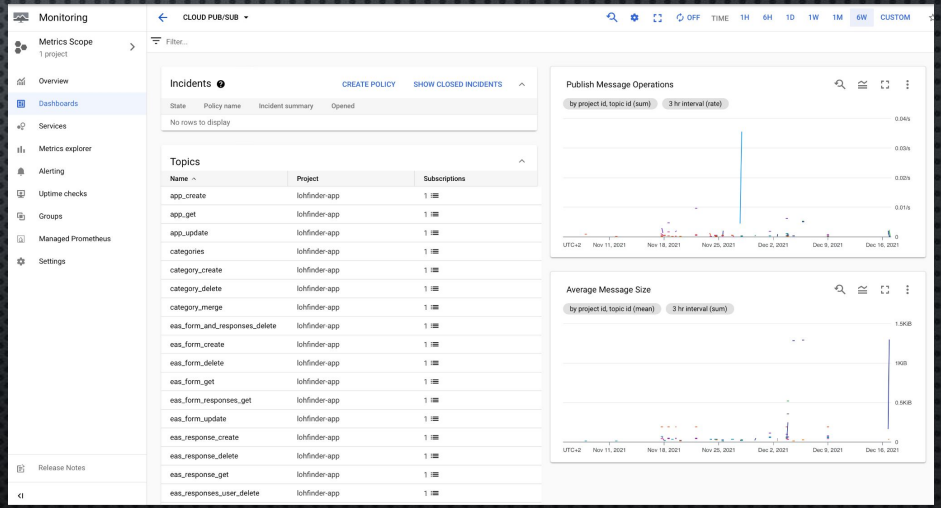
GKE provides **Cloud operations for GKE** a dashboards that show clusters and services health, list of errors, CPU utilization, etc.



Dashboard example

Monitoring | Pub/Sub

Google Pub/Sub provides a dashboard for each topic emitted and event subscriptions. It also allows to measure topic delivery health.



Dashboard screenshot