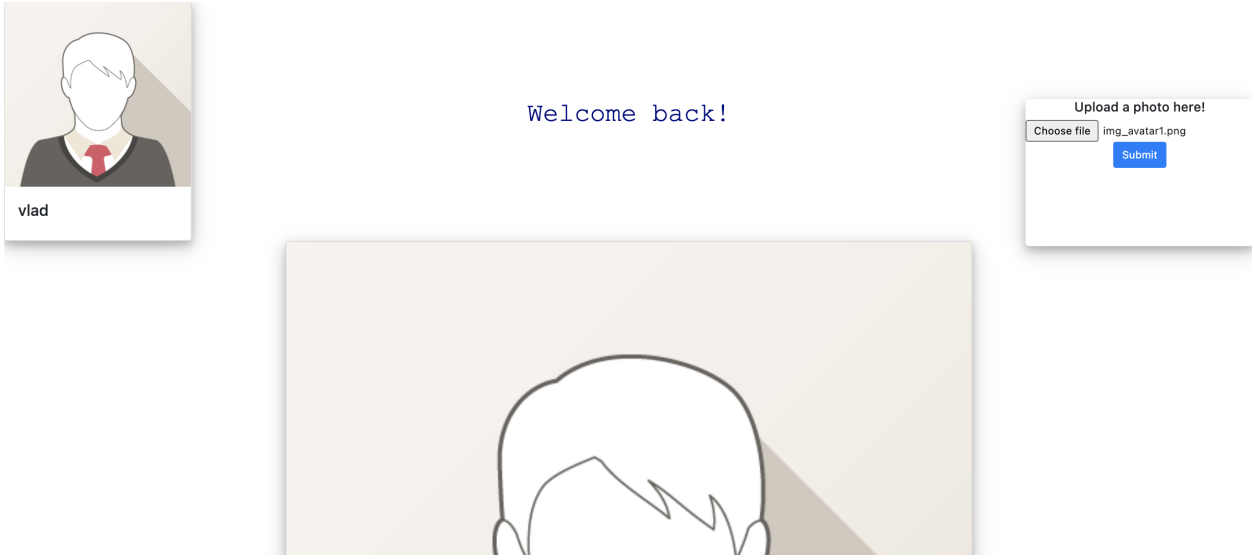
Primire poze cu websockets, upload de poze in fass si autentificare

User Interface

Partea de UI a fost creata doar cu ajutorul urmatoarelor limbaje: HTML, CSS si Javascript, dar si cu putin ajutor din partea Bootstrap-ului. Ea a fost gandita pentru a afisa pe partea centrala cate o poza, fiind astfel reponsive pe orice platforma. In partea din stanga sus este afisat profilul utilizatorului, compus dintr-o poza si username-ul acestuia. In partea opusa, in coltul din dreapta sus este formularul ce permite postarea unei imagini catre server. Si nu in ultimul rand, pe centru in partea de sus este un mesaj de bun venit adresat utilizatorului. UI-ul se poate vedea in urmatoarea imagine:



AJAX Request

Pentru a posta o imagine catre FaaS (folosit pentru adaugarea acesteia in Firebase, alaturi de redimensionarea ei si de adaugarea altor date) , am folosit urmatoarea metoda: a fost implementat un formular HTML(1), iar evenimentul ce posteaza formularul a fost prins si redirectionat catre API-ul din Cloud cu ajutorul unei cereri AJAX(2), dupa cum urmeaza:

(1) Formularul HTML

```
        <form id='uploadImage' action=''>
            <h5>Upload a photo here!</h5>
            <input class='form-contorol' type="file" id="myFile"
name="image">
            <input class='btn btn-primary' type="submit">
        </form>
```

(2) AJAX Request

```
$(document).ready(function (e) {
  $('#uploadImage').on('submit', (function (e) {
    e.preventDefault();
    var formData = new FormData(this);
    formData.append('user', getUser());
    alert('catched!');
    $.ajax({
      type: 'POST',
      url:  < OUR API URL>,
      data: formData,
      cache: false,
      contentType: false,
      processData: false,
      success: function (data) {
        console.log('success');
        console.log(data);
      },
      error: function (data) {
        console.log('error');
        console.log(data);
      }
    });
  }));

});
```

WebSockets

WebSockets au fost folositi pentru a notifica utilizatorii conectati cu referire la aparitia unei noi poze. Poza noua poate fi incarcata de catre oricare utilizator. Pe partea de client, la accesarea paginii de HOME, se creeaza o conexiune automata intre WebSocket-ul de pe partea de client cu cel de pe partea de server. WebSocket-ul de pe partea de server efectueaza periodic o interogarea a pozelor din Cloud, iar daca acestea sunt mai multe decat cele de la ultima interogarea, trimite o notificare catre toti utilizatorii, urmand ca acestia sa reincarce toate imaginile.

Arhitectura

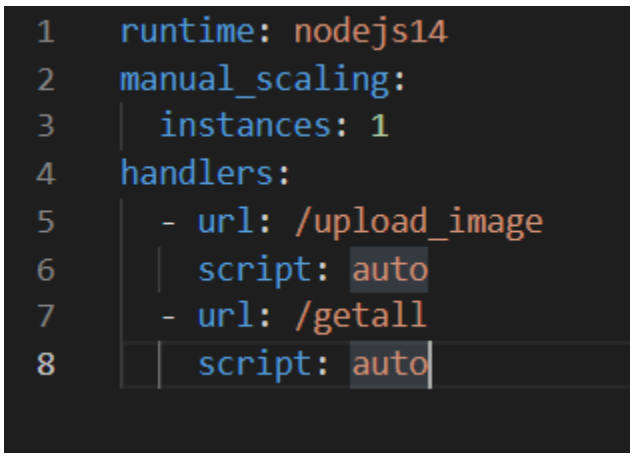
API
Serveste UI-ul, imaginile, lista de imagini si ofera functionalitate de websockets pentru a notifica clientii de noi imagini. Foloseste Firebase pt metadata, Cloud storage pentru imagini si va fi in Google Cloud.

FAAS (client -> faas)
Upload de poze compresate in Cloud Storage pentru a distribui long running connections in mai multe instante.

OnPremise
User auth folosind un server de OAuth.

Google Cloud

Stocarea imaginilor se realizeaza folosind Google Cloud, si anume serviciile Storage si Firestore.
In storage a fost creat un bucket in care se salveaza imaginea propriu-zisa, avand ca denumire un id format din cifre. Metadatele sunt salvate in Firestore ca si documente avand ca id numele imaginii, iar ca detalii utilizatorul care a incarcat imaginea si data uploadului.
Pentru managementul imaginilor se folosesc functii http ce au fost deployed in Cloud.
Functia GET /getall returneaza id-urile tuturor documentelor din Firestore si cu acestea sunt downloadate imaginile din storage si afisate in index.html.
Functia POST /upload_image preia o imagine locala, o incarca in bucket si creeaza un nou document in Firestore pt aceasta.
Deploymentul se realizeaza cu ajutorul unui fisier app.yaml ce contine informatiile necesare, apoi ruland comanda *gcloud app deploy*.



Fisierul app.yaml