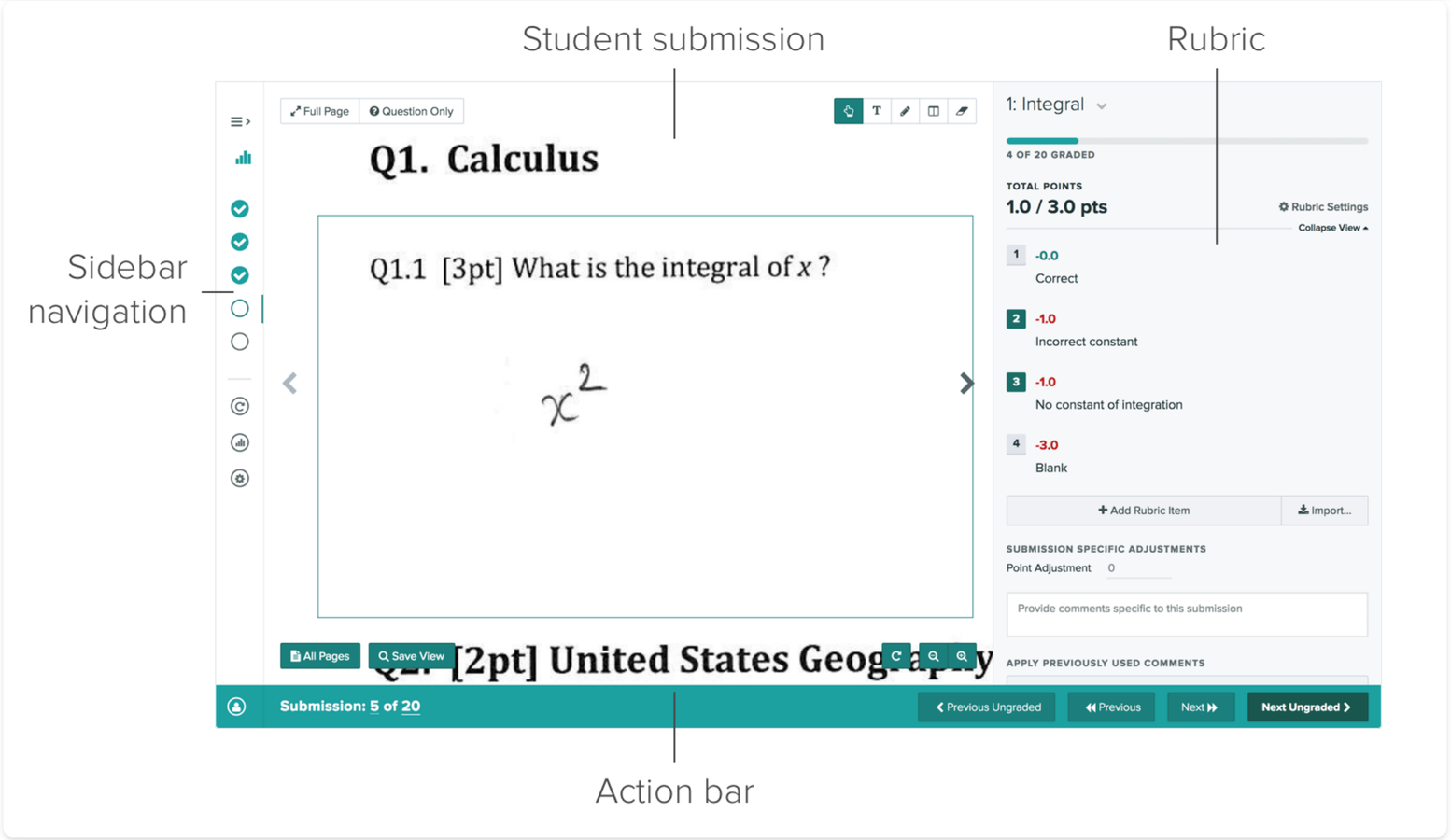
Correct Exam

Modern software architecture in practise

Press Space for next page →

Highly inspired by GradeScope Solution

> Gradescope grading software allows students to receive faster and more detailed feedback on their work, and allows instructors to see detailed assignment and question analytics. It is an easy way to take submissions digitally in order to preserve the original work and allow for quick and easy viewing from anywhere.



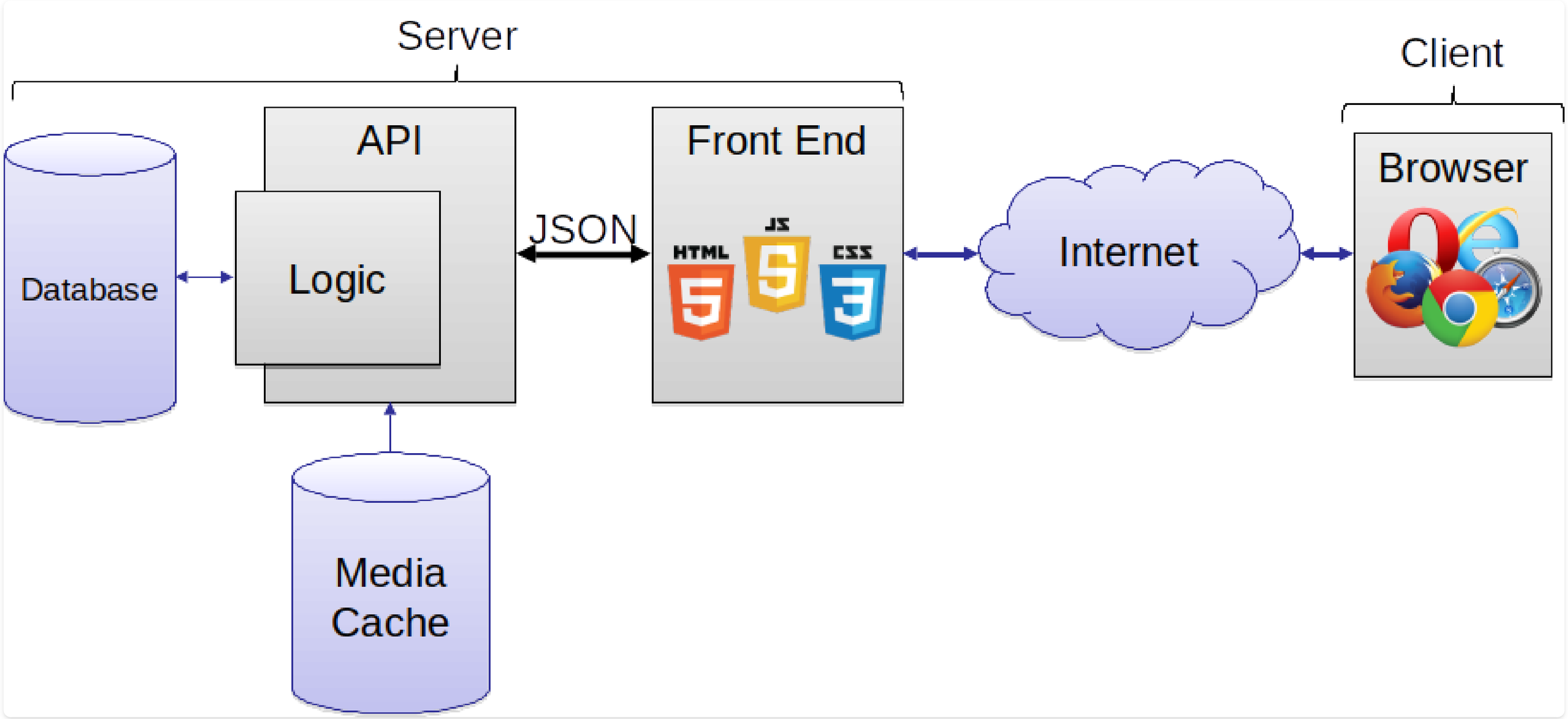
Why building that piece of software ?

- Enable to correct exams during meetings 😀
- Try to initiate a project with Younup to see how we could engage efficiently software engineer available for some days
- Create an open source implementation of real software with complex architecture to have a case study for
  - experiments in software engineering research
  - explaining modern software architecture to students

The technical architecture

- **Quarkus** for the back (Java + native compiltation through GraalVM)
- **Angular 13.3** for the front
  - **pdf.js** to play with pdf (exam, scan exam, feedback for students)
  - **fabric.js** to draw on top of a pdf
  - **opencv** in wasm within a web worker to analyse the scan
  - **tensorflow JS** with the browser for digit and letter recognition
  - …
- **Docker** and K8S to deploy the back and the monitoring layer
- Front is hosted in a CDN to follow the JamStack architecture (currently github page)
- CI/CD using **github action**, **dockerhub webhook**, and **gowebhook**

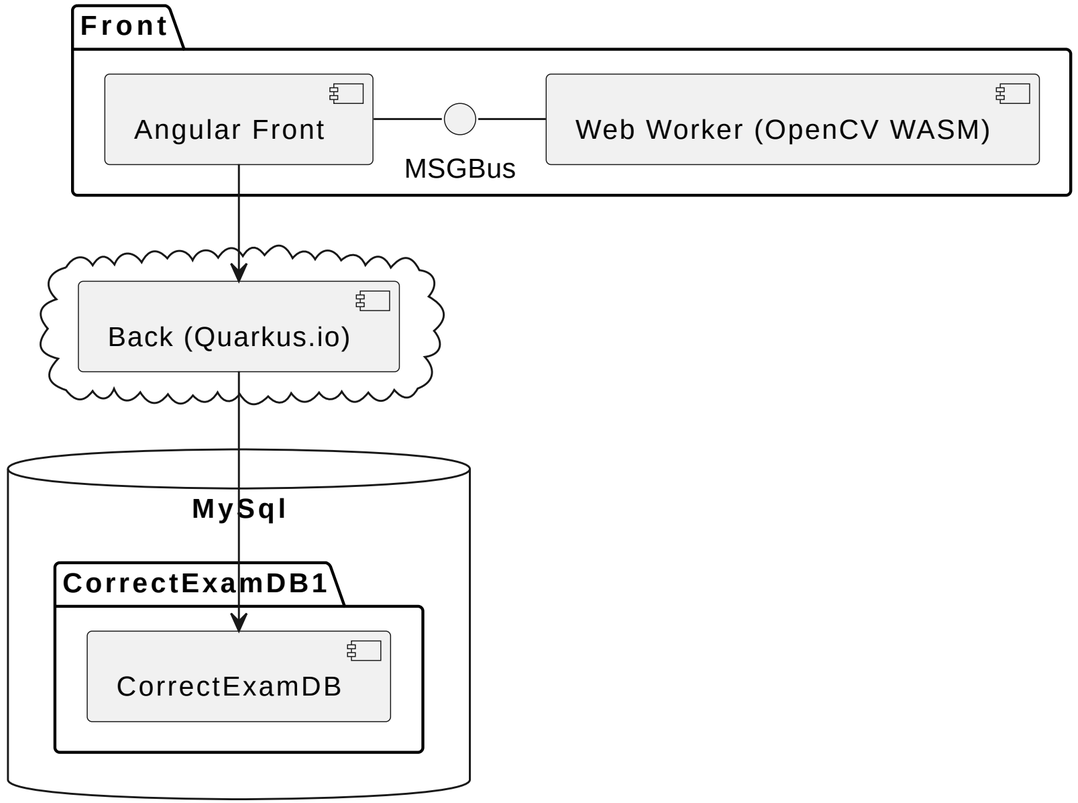
Architecture overview



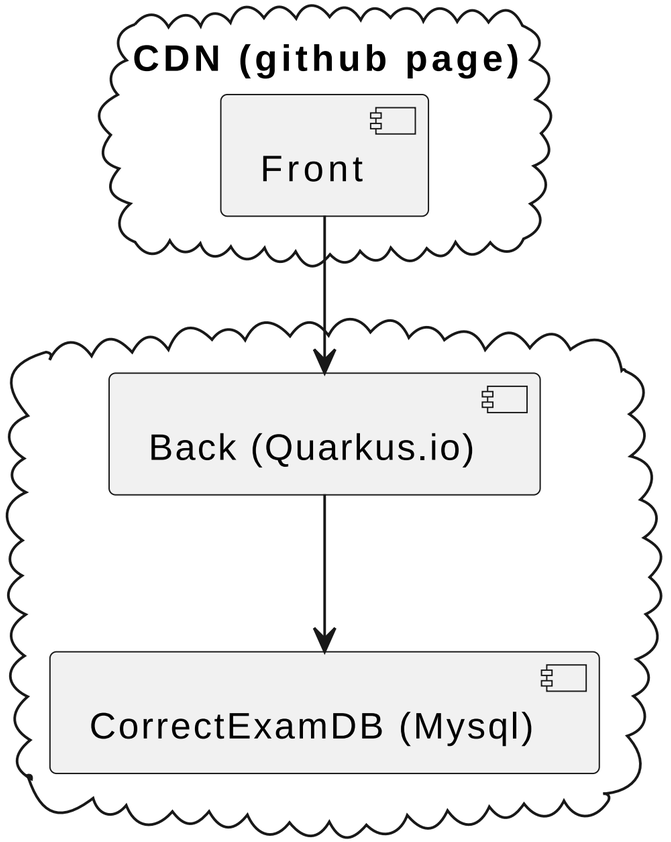
Diagrams

You can create diagrams / graphs from textual descriptions, directly in your Markdown.

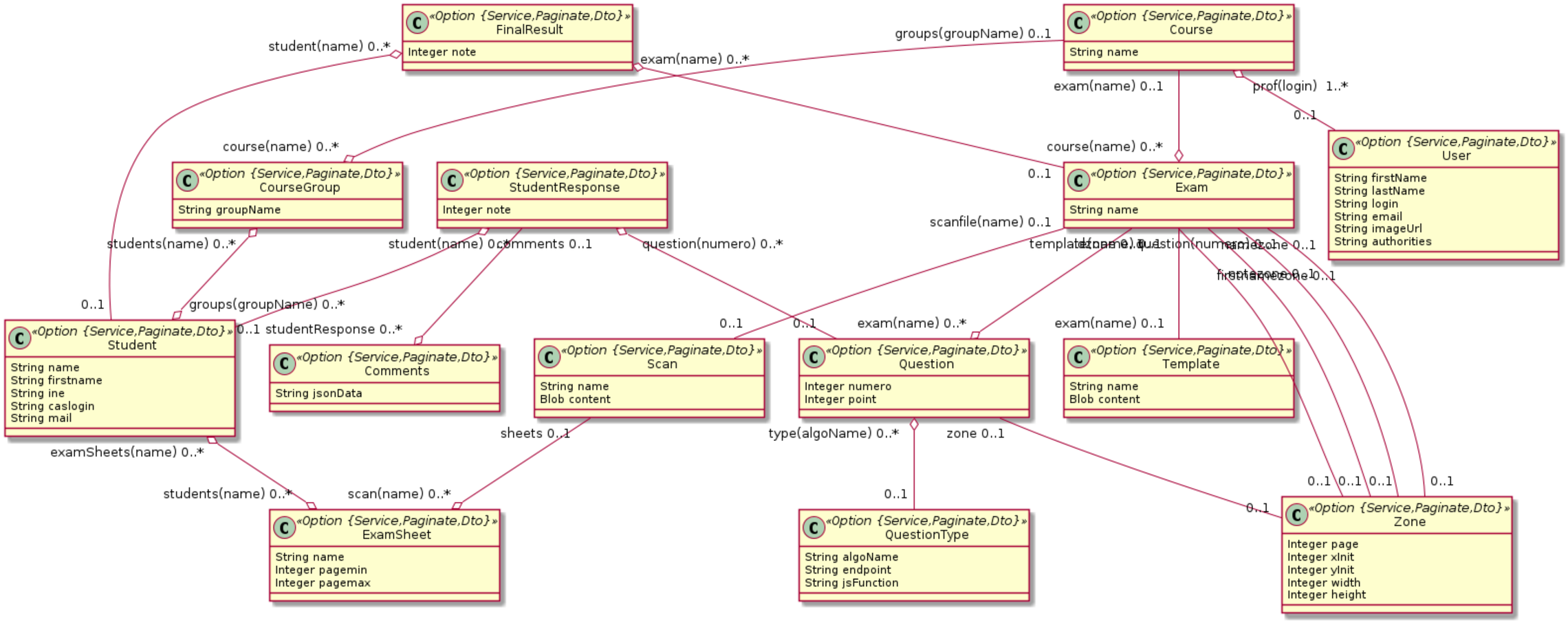
Conceptual architecture



deployed architecture



Business model



JHipster to generate the application skeleton

> JHipster is a development platform to quickly generate, develop, & deploy modern web applications & microservice architectures.

- jhipster 7.7.0 for the front
- jhisper 6.10.15

> Manual alignement of the API evolution 😀 (@Djamel)

Where we are

Done

- [x] Vue home
- [x] Donnée factice
- [x] Afficher vraie liste de cours (requête au back)
- [x] Masquer vue entité pour user
- [x] Premère page
- [x] Composant Cours viusualisation
- [x] Composant Insérer groupe étudiant
- [x] Composant supprimer UE
- [x] Voir groupe étudiant pour une UE incluant VIDER LISTE ETUDIANT pour cet UE
- [x] Refactoring module
- [x] Composant Créer exam

Done

- [x] Composant ExamEditerzone
  - [x] Créer type de rectangle fabric
- [x] Vérifier compilation native
- [x] Vue question
- [x] Mise en place CI
- [x] Mise en place CD (front)
- [x] Vue home admin
- [x] Bouton retour dans la partie annoter document
- [x] Bouton charger scan
- [x] Bouton associé copies
- [x] Bouton Corriger
- [x] Intégration opencv
- [x] Test intégration opencv
- [x] Mise en place CD (back)

Ongoing

- [ ] Backup alignement pdf
- [ ] Découpe numéro (bounding box, opencv)
- [ ] Association étudiants (angular)
- [ ] Icone verte dans le workflow de correction

TODO

- [ ] TensorFlow.js, reconnaissance chiffre (match étudiant)
- [ ] Bouton voir et exporter résultat
- [ ] Bouton envoyer résultats aux étudiants
- [ ] Bouton voir copie corrigée
- [ ] Bouton associé copies
- [ ] Bouton Corriger
- [ ] Bouton voir et exporter résultat
- [ ] Bouton envoyer résultats aux étudiants
- [ ] Bouton voir copie corrigée
- [ ] Traduction title route
- [ ] Traduction composant module scanexam
- [ ] Retravailler traduction
- [ ] Mettre à jour la vue du numéro de question quand mise à jour dans le menu propriété.

TO Think

- [ ] Réfléchir barème positif ou négatif
- [ ] Réfléchir annotation automatique par question
- [ ] Module alignement image
- [ ] Module reconnaissance nom

To fix

- [ ] Configuration email back
- [ ] Remove ununsed dependecies (crop ngx, opencvngx)

Fixed

- [x] Exam
- [x] Question
- [x] ExamSheet
- [x] Final Result
- [x] Student Response
- [x] Comments
- [x] opencv path in the deployed version

Demo and code overview

- demo
- repo back
- repo front

A simple RPC to manage the communication between openCV worker and Angular

In the worker

```
addEventListener('message', e => {
  switch (e.data.msg) {
    case 'hello': {
      const response = `worker response to ${e.data.msg}`;
      postMessage({ msg: response, uid: e.data.uid });
      break;
    }
    ...
```

In the service

```
ready!: Promise<void>;
 subjects = new Map<string, Subject<any>>();
 constructor() {
   if (typeof Worker !== 'undefined') {
     // Create a new
     this.worker = worker;
     this.worker.onmessage = ({ data }) => {
       if (this.subjects.has(data.uid)) {
         this.subjects.get(data.uid)?.next(data.payload);
         this.subjects.get(data.uid)?.complete();
       }
     };
     this.worker.onerror = e => {
       if (this.subjects.has((e as any).uid)) {
         this.subjects.get((e as any).uid)?.error(e);
       }
     };
     const p = new Subject();
     this.subjects.set('0', p); // Ask to load opencv wasm to the worker
     this.ready = new Promise((res, rej) => {
       this.subjects
         .get('0')?.asObservable().subscribe(() => {res();},() => rej());
     });
```

In the service

```
 private _dispatch<T>(msg1: any, pay: any): Observable<T> {
  const uuid1 = uuid(); // ⇨ '9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d'
  this.ready.then(() => {
    this.worker.postMessage({ msg: msg1, uid: uuid1, payload: pay });
  });
  const p = new Subject<T>();
  this.subjects.set(uuid1, p);
  this.ready = new Promise((res, rej) => {
    this.subjects.get(uuid1)?.asObservable().subscribe(() =>
        {res();},
        () => rej()
      );
  });
  return p.asObservable();
}
public imageProcessing(image: ImageData): Observable<ImageData> {
  return this._dispatch('imageProcessing', image);
}
```

Learn More

Documentations · GitHub · Showcases