TECHNOLOGIES WEB

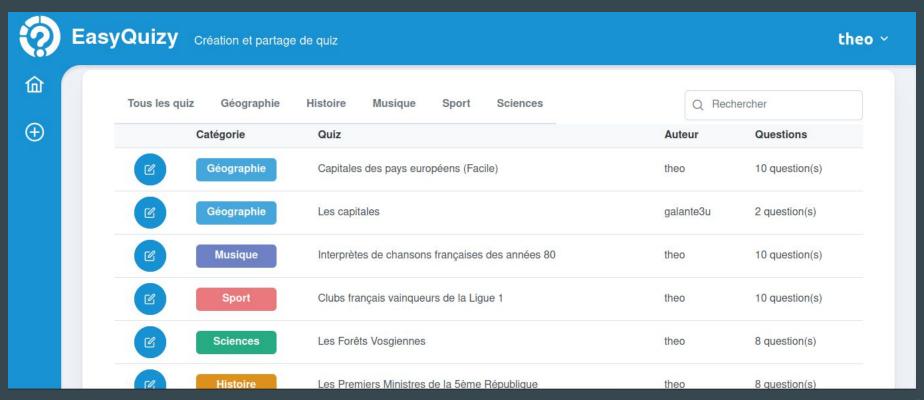


Théo JOFFROY Matthieu GALANTE



EasyQuizy

Site de création et de partage de quiz





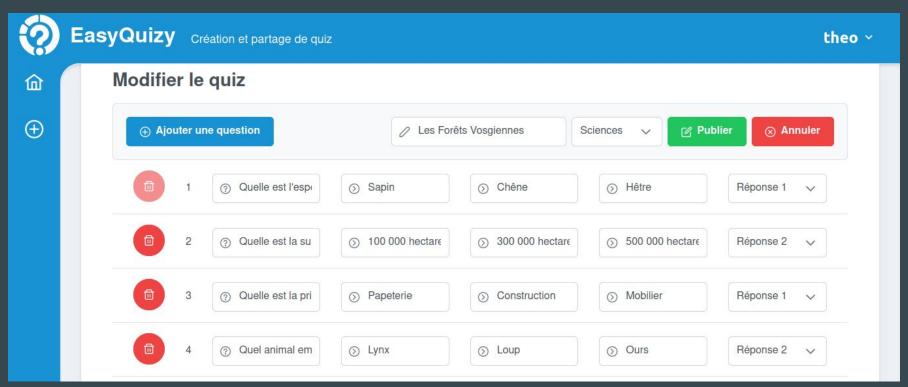






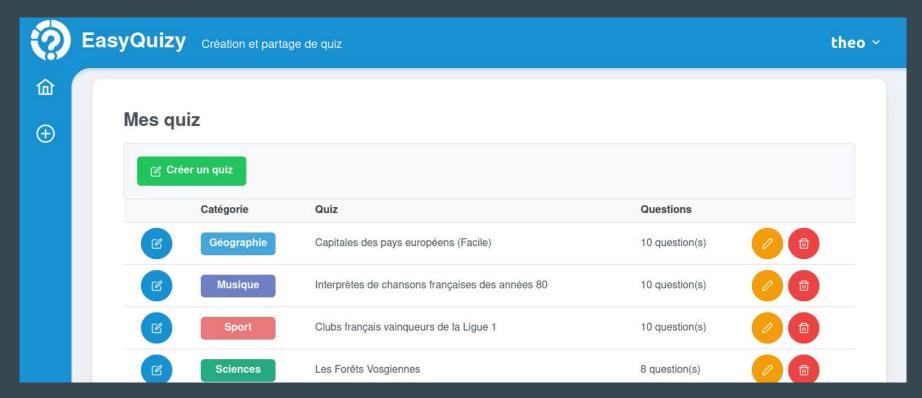


Vues Création / Modification d'un quiz





Vues Gestion des quiz publiés





→ /user GET POST

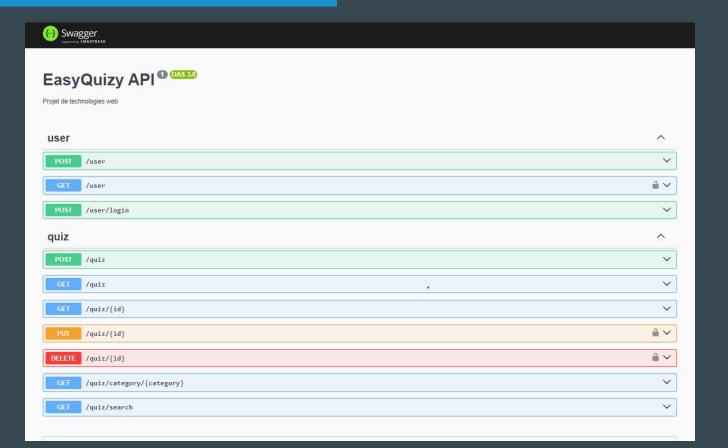
→ /user/login POST

- → /quiz GET POST
- → /quiz/:id

 GET PUT DELETE
- /quiz/category/:category
 GET
- → /quiz/search?keyword= GET



Documentation





Détails techniques



Composants PrimeNG





Détails techniques

Validations I/O

```
@IsString(
@IsNotEmpty(
readonly title : string;
@ApiProperty({
   name: 'category',
   description: 'Category of the quiz',
    example: '2'.
@IsNumber()
readonly category : number;
@ApiProperty({
   name: 'questions',
    description: 'List of the questions',
    type: [QuestionDto],
@IsInstance(QuestionDto, { each: true })
@ValidateNested({ each: true })
@Type(() => QuestionDto)
readonly questions : QuestionDto[]
```

```
@ApiProperty({
   name: 'id',
   description: 'Unique identifier in the database',
    example: '5763cd4dc368a38acd387737',
@Expose()
@Type(() => String
id : string:
@ApiProperty({
   name: 'author',
    description: 'Author of the quizz',
    example: 'UserTest',
@Expose()
@Type(() => String)
author: string;
@ApiProperty({
    name: 'title'.
   description: 'Title of the quizz',
    example: 'Le super quiz',
@Expose |
```



Difficultés rencontrées

Tester le client sans que les endpoints soient mis en place

→ Mock du backend avec JSON Server

```
server.get('/quiz/category/:category', (req, res) => {
  res.status(200).send(quizData.quiz.filter(q => q.category.toString() === req.params.cat
server.put('/quiz/:id', (req, res) => {
  const bearerId = loginData.users.find(user => user.token === req.headers.authorization.
  if(quizData.quiz.find(q => q.id.toString() === req.params.id) === undefined){
    res.status(404).send():
  }else if(bearerId === undefined || bearerId !== quizData.quiz.find(q => q.id.toString()
    res.status(401).send();
    quizData.quiz = quizData.quiz.map(q => q.id.toString() === req.params.id ? req.body :
    res.status(200).send();
server.delete('/quiz/:id', (reg, res) => {
  const bearerId = loginData.users.find(user => user.token === req.headers.authorization...
  if(quizData.quiz.find(q => q.id.toString() === req.params.id) === undefined){
    res.status(404).send();
```

```
"quiz":
    "id": 0.
   "author": "UserTest1",
    "title": "Test Ouiz 1",
    "category": 1,
    "questions":
        "question": "Question 1",
        "answers": [
          "A",
          "B",
        "correctAnswer": "2"
        "question": "Question 2",
        "answers":
          "D",
        "correctAnswer": "0"
```



Difficultés rencontrées

Authentification avec JSON Web Token (JWT)

- → Token dans le header Authorization
- → Protection des routes avec JwtAuthGuard

```
@ApiBearerAuth()
@UseGuards(JwtAuthGuard)
@Put(':id')
updateQuizzById(@Param() params: HandlerParamsId, @Body() creat
    return this._quizzService.modify(params.id, createAndPutQui
}
```



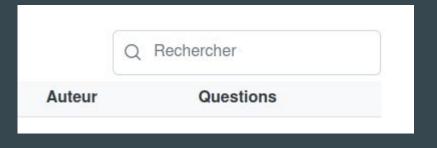
Difficultés rencontrées

→ Indexation avec

MongoDB pour effectuer

des recherches par

mot-clés



→ Protocole HTTPS pour protéger les informations de connexion



Démonstration