

TITRE PROFESSIONNEL DÉVELOPPEUR WEB ET WEB MOBILE

Soutenance

Bernard ARROUES – Tout'O'Poils

Paris - 31/03/2023

Qui suis-je?

Parcours

- *effacé*
 - 2015 – 2017
- *effacé*
 - 2017 – 2020
- O'clock
 - décembre 2022 – février 2023
 - Spécialisation back-end

Pourquoi ?

- Intérêt depuis tout jeune pour les nouvelles technologies
- La programmation en hobby
- Lassitude + covid = changement de voie

SOMMAIRE



I. ÉQUIPE ET
PRÉSENTATION DU
PROJET



II. CONCEPTION



III. DÉVELOPPEMENT



IV. RÉALISATIONS
PERSONNELLES



V. DÉMONSTRATION
DU PROJET



VI. EXEMPLE DE
RECHERCHE



VII. CONCLUSION

Notre équipe



Angélique
Product owner
Scrum master
Dev back



Luis
Dev front



Mathilde
Dev front



Denise
Dev back







Git master
Dev back

Tout'O'Poils - Genèse

CONSTAT

les antennes locales de la SPA manque d'outils de suivi efficace des animaux.

AUDIENCE

-  Antennes locales de la SPA
-  Bénévoles
-  Employés
-  Refuges animaliers

Organisation du travail

- **4 sprints**, du jeudi au jeudi
- Utilisation de Trello, communication via Discord
- Daily meeting à 9 heure
- Journée type de **9h à 18h**, possibilité de travailler seul hors horaires

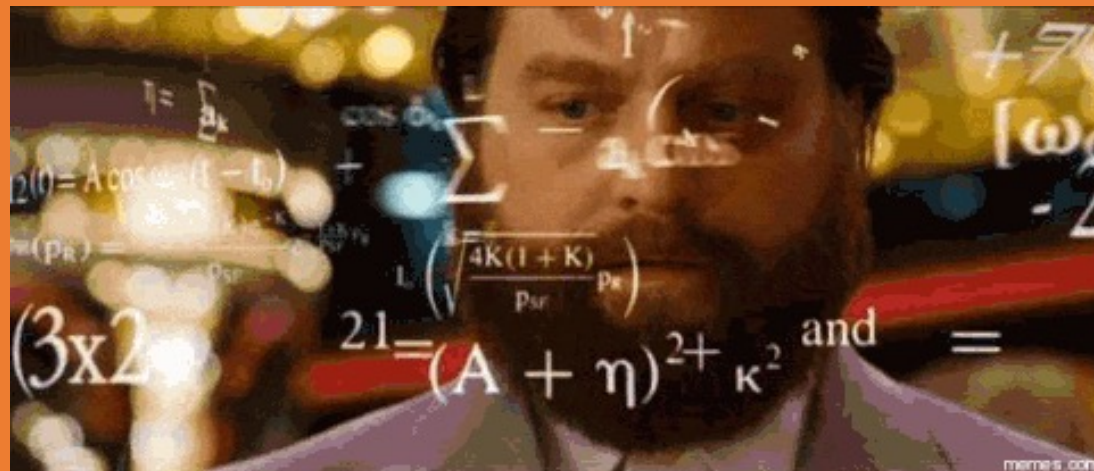


- User stories
- Wireframe
- Réflexions sur le MVP
- Etc...

- Fix des bugs restants
- Ajout de petites fonctionnalités

Merge sur la branche **main** et déploiement le **jeudi midi**

CONCEPTION



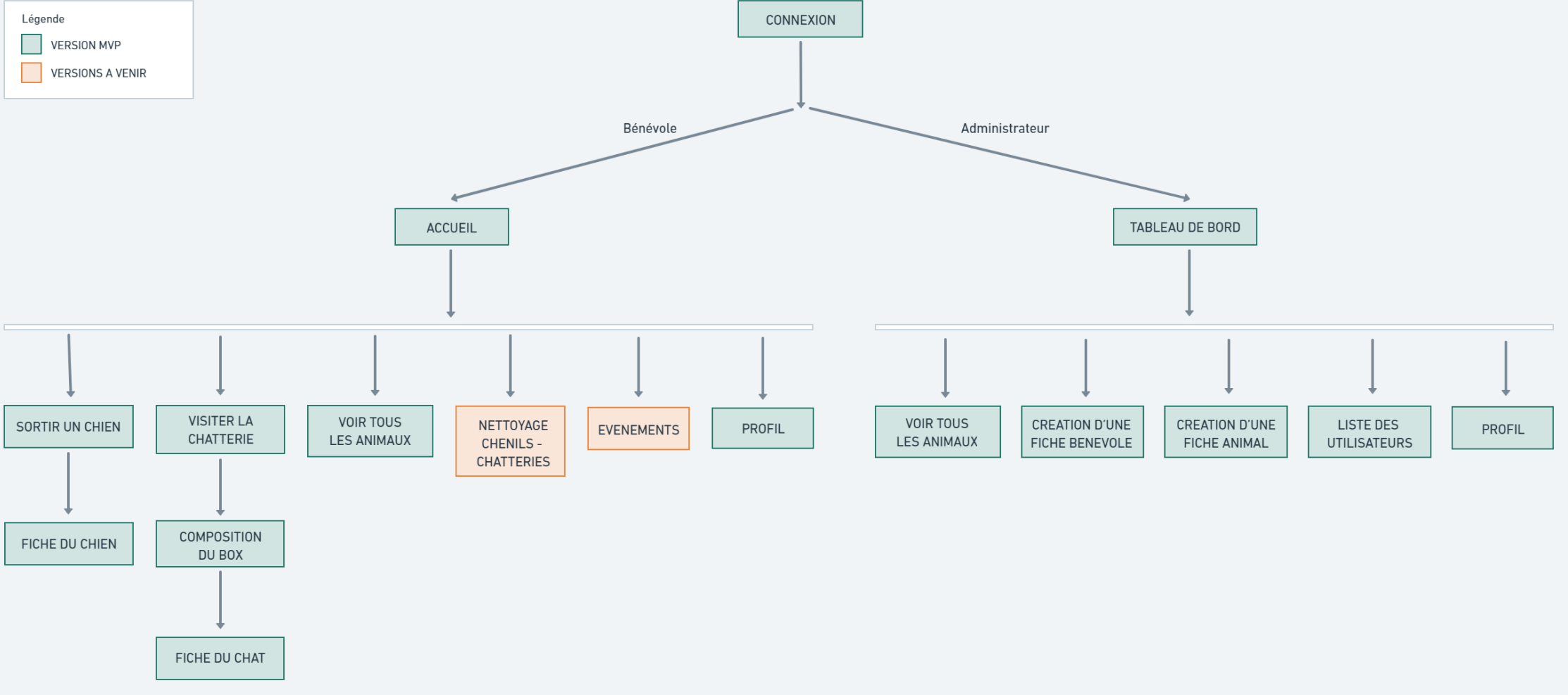
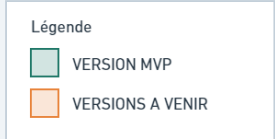
User stories

- 3 types d'utilisateurs
 - Visiteur
 - Bénévole
 - Administrateur (= employé de la SPA)
- En vert: **MVP**
- En jaune: versions futures

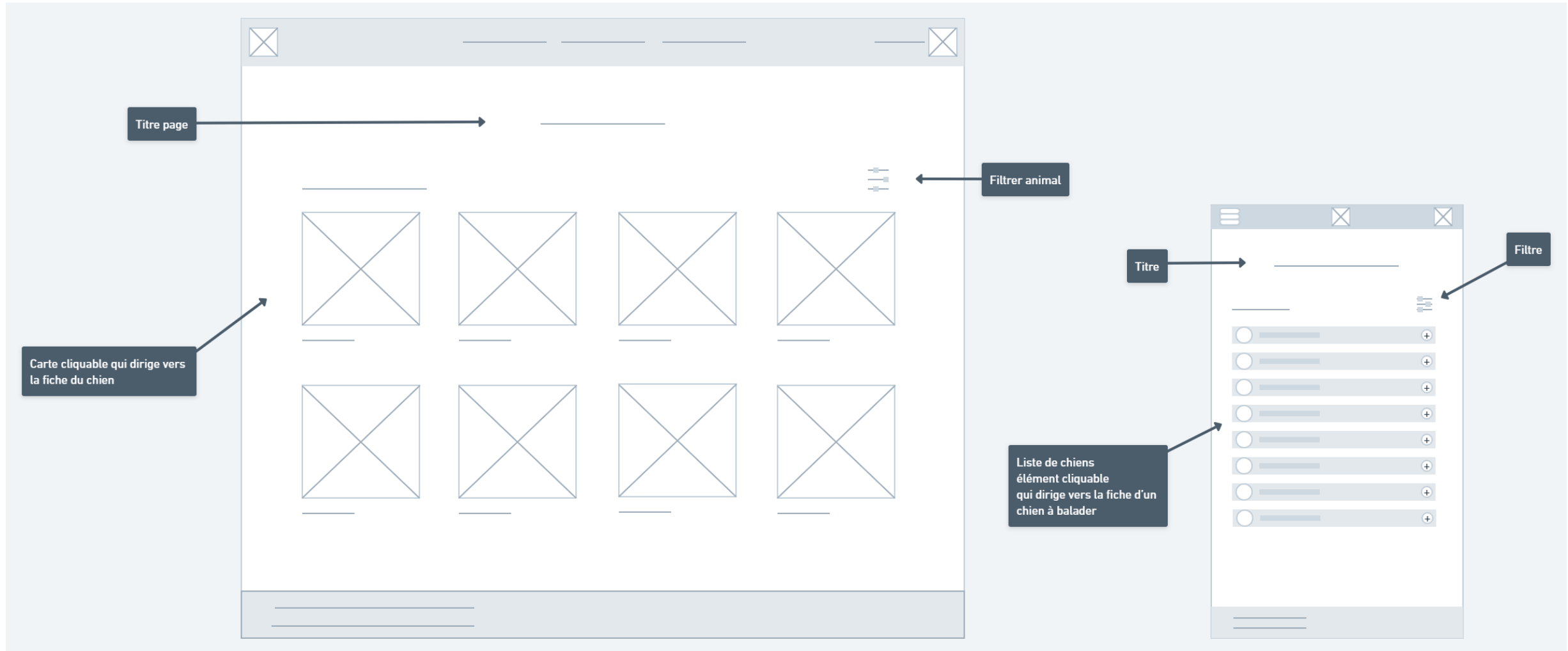
En Tant que	Je souhaite	Afin de
visiteur	me connecter à l'appli	accéder aux différents services
visiteur	me déconnecter de l'appli	pour laisser la place à d'autres utilisateurs
utilisateur connecté	accéder à mon profil	modifier mes informations personnelles
bénévole	voir la liste des chiens	voir ceux qui doivent être sortis
bénévole	connaître les tâches prioritaires	d'apporter mon aide rapidement
bénévole	pouvoir filtrer les caractéristiques de l'animal	choisir l'animal correspondant à mes préférences
bénévole	savoir où je peux voir les chats	leur rendre visite et jouer avec
bénévole	consulter la liste des animaux	pour voir leur fiche
bénévole	savoir si un animal précis a été adopté	me renseigner sur mon chouchou (favoris)
bénévole	signaler un problème	informer au plus vite les référents spa
bénévole	savoir si on a besoin de moi pour une collecte ou autre	d'apporter mon aide
administrateur	savoir l'état de sortie des chiens	savoir si tous les chiens sont sortis
administrateur	ajouter un animal dans le refuge	pour qu'il puisse avoir un suivi

Extrait des user stories

Arborescence

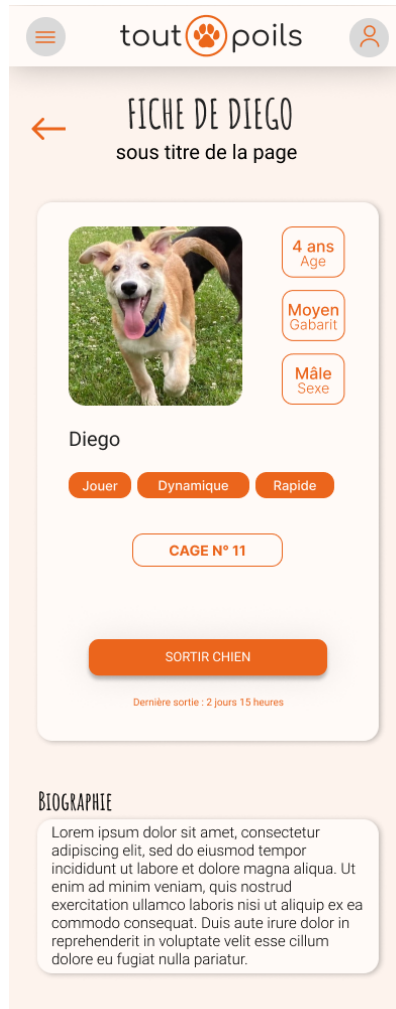


Wireframes

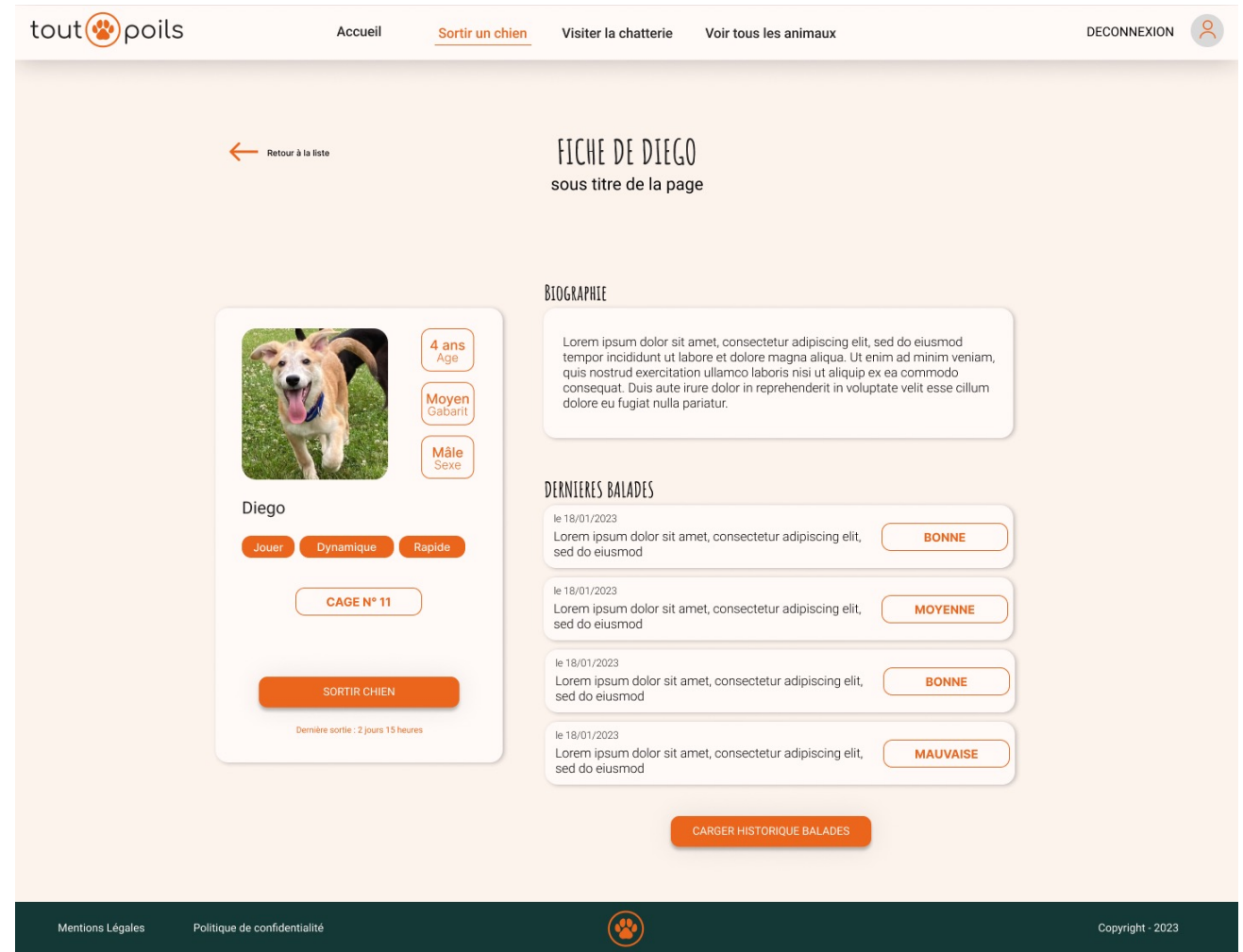


Extrait des wireframes

Maquettes

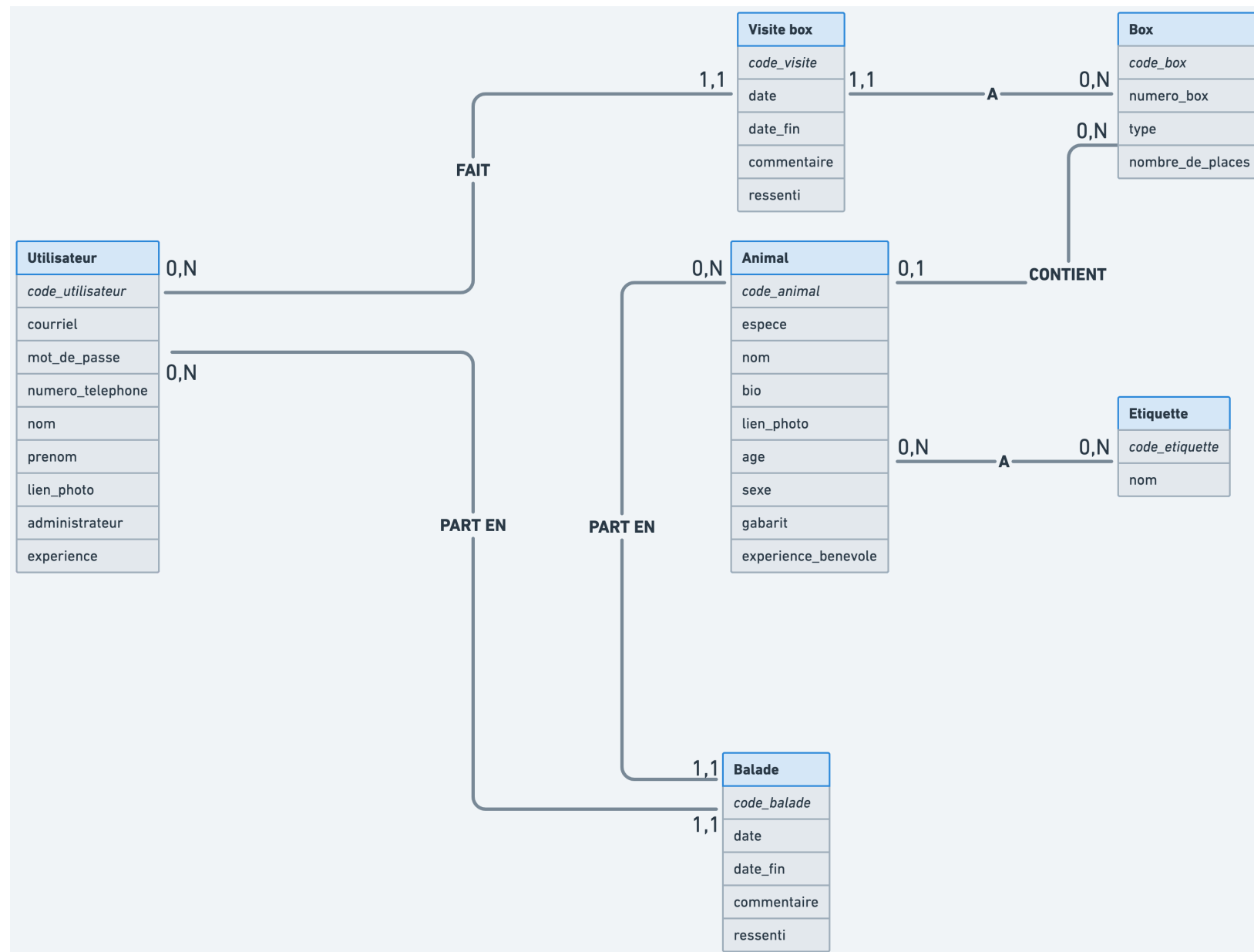


Mobile (rogné)

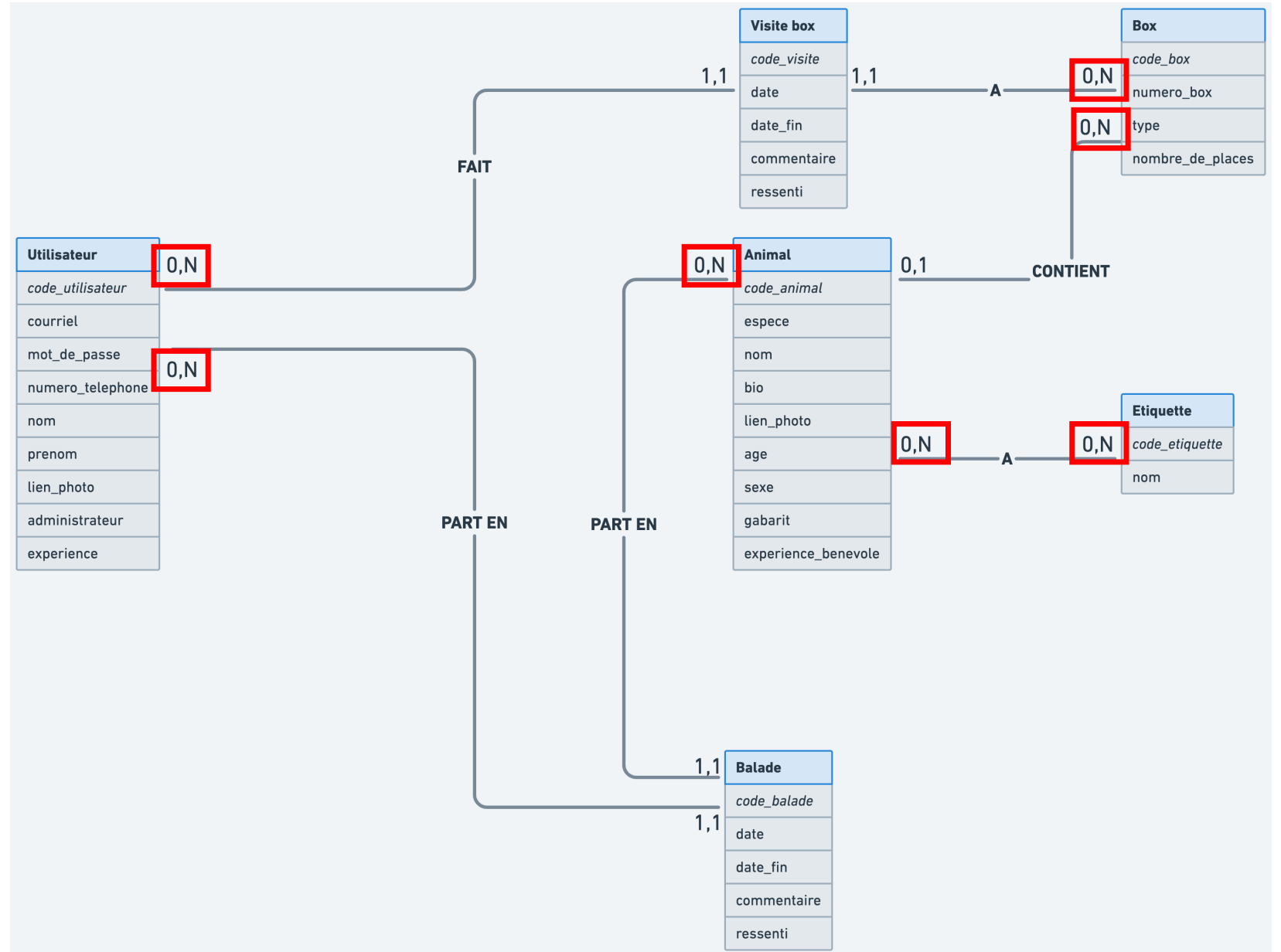


Desktop

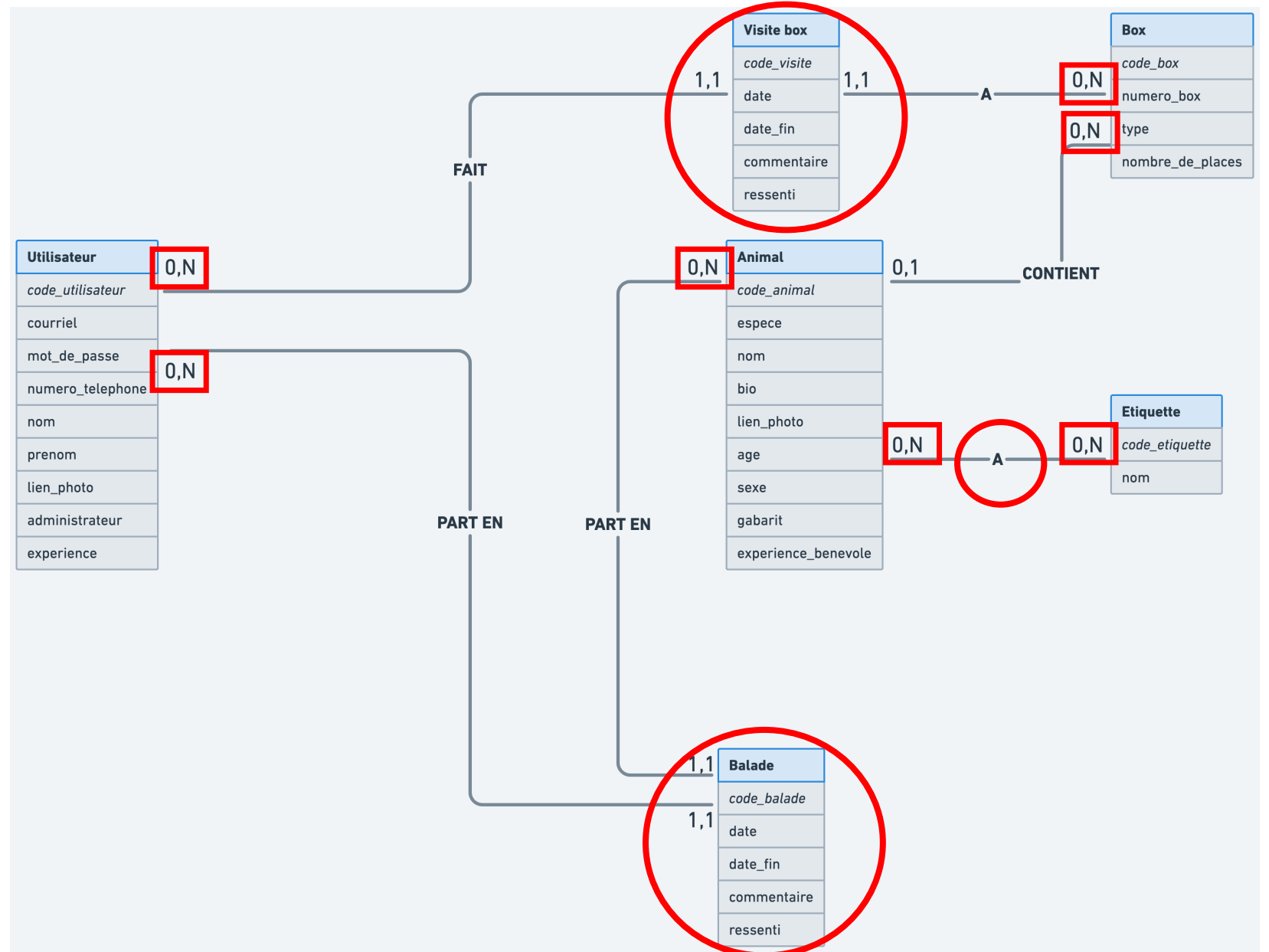
MCD



MCD



MCD



MLD

user (id, email, password, name, firstname, phone_number, url_image, admin, experience)

box (id, type, number, nb_of_places)

animal (id, species, name, url_image, age, bio, gender, size, volunteer_experience, **#box_id**)

walk (id, date, end_date, comment, feeling, **#user_id**, **#animal_id**)

tag (id, name)

animal_has_tag (**#animal_id**, **#tag_id**)

visit (id, date, end_date, comment, feeling, **#user_id**, **#box_id**)

MLD

user (id, email, password, name, firstname, phone_number, url_image, admin, experience)

box (id, type, number, nb_of_places)

animal (id, species, name, url_image, age, bio, gender, size, volunteer_experience, **#box_id**)

walk (id, date, end_date, comment, feeling, **#user_id**, **#animal_id**)

tag (id, name)

animal_has_tag (**#animal_id**, **#tag_id**)

visit (id, date, end_date, comment, feeling, **#user_id**, **#box_id**)

DICTIONNAIRE DES DONNEES

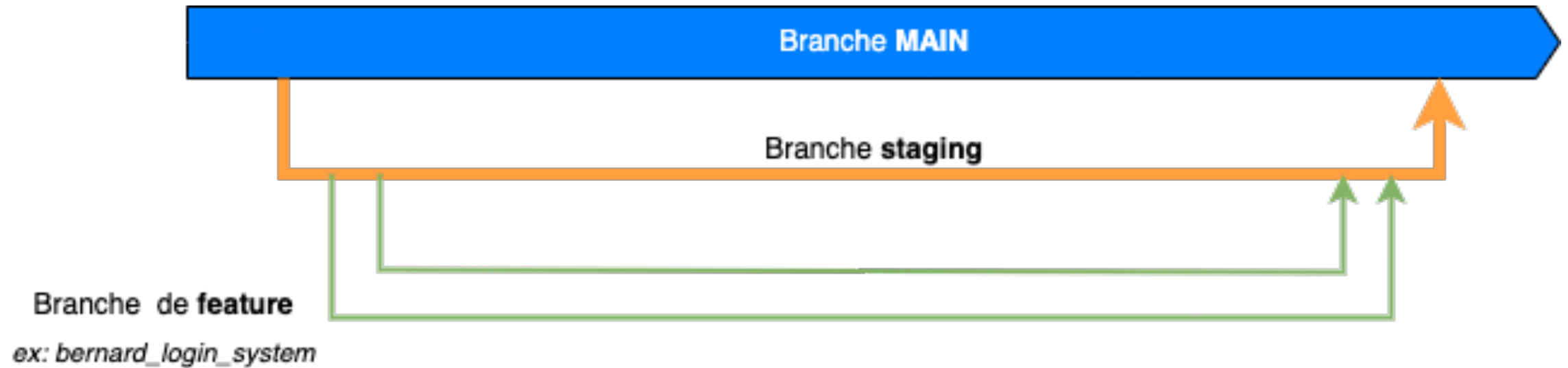
Nom de la donnée	Désignation	Type	contrainte
code_utilisateur	Code numérique d'un utilisateur	integer	generated as identity primary key
courriel	Adresse email d'un utilisateur	text	not null
mot_de_passe	Mot de passe d'un utilisateur	text	not null
numéro_telephone	Numéro de téléphone d'un utilisateur	text	
nom	Nom de l'utilisateur	text	not null
prenom	Prénom de l'utilisateur	text	not null
lien_photo	Url vers la photo de l'utilisateur	text	
administrateur	Confirmation du statut admin de l'utilisateur	booléen	not null, default false
experience	Niveau d'expérience de l'utilisateur	text	not null check ("beginner", "medium", "expert"), default ("beginner")
code_animal	Code numérique d'un animal	integer	generated as identity primary key
espece	Espèce de l'animal	text	not null check("cat", "dog", "other")
nom	Nom de l'animal	text	not null
bio	Résumé de l'animal	text	
lien_photo	Url vers la photo de l'animal	text	
etiquette	Etiquette de la caractéristique de l'animal	text	not null
age	Age de l'animal	timestampz	
sexe	Sexe de l'animal	text	check ("male", "female")
gabarit	Gabarit de l'animal	text	check ("small", "medium", "big")
Expérience_benevole	Expérience demandée pour s'occuper de l'animal	text	not null default 'beginner'

Extrait du dictionnaire des données

Stack technique

FRONT	BACK	DEVELOPER EXPERIENCE
<ul style="list-style-type: none">- React- Bootstrap- Sass- Redux	<ul style="list-style-type: none">- Express- Prisma- PostgreSQL- Envoi de mail:<ul style="list-style-type: none">- Nodemailer- SendInBlue (fournisseur SMTP)- Handlebars (templates)- Upload d'images:<ul style="list-style-type: none">- Multer- AWS S3- SDK S3	<ul style="list-style-type: none">- Github- Eslint- Prettier

Gitflow



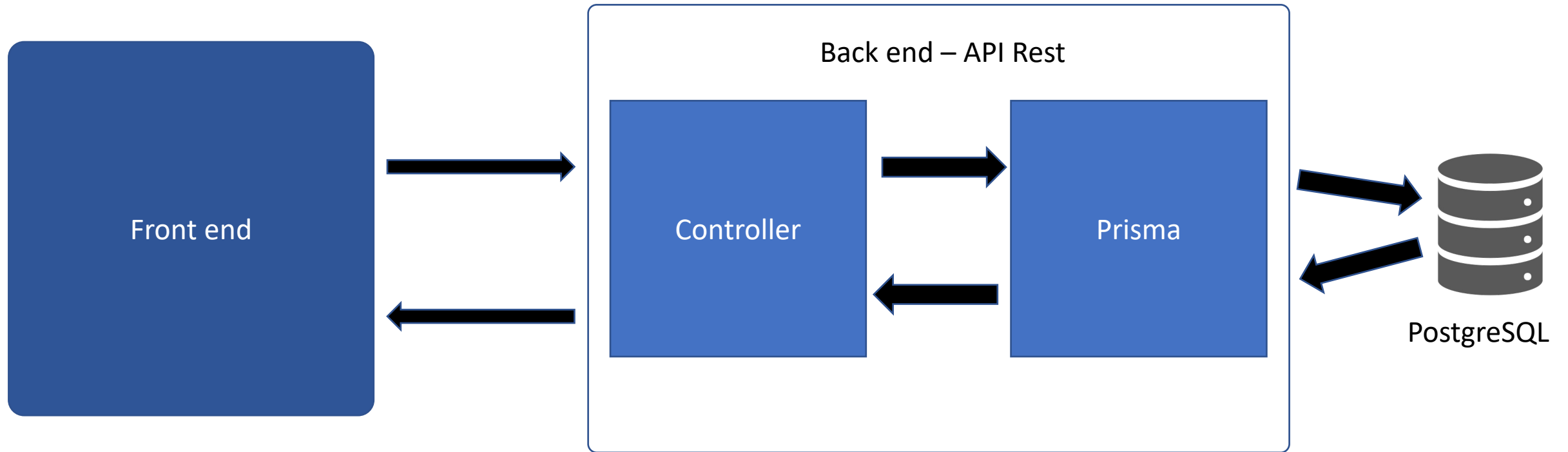
- Branche de **feature**: contient les features sur lesquelles les membres de l'équipe travaillent
- Branche **staging**: merge des branches de feature, réalisation d'essais
- Branche **main**: merge de la branche staging, version de l'application en production

Le merge ne se fait que par le biais d'une **Pull Request**, et revue de code par le Git Master

DÉVELOPPEMENT



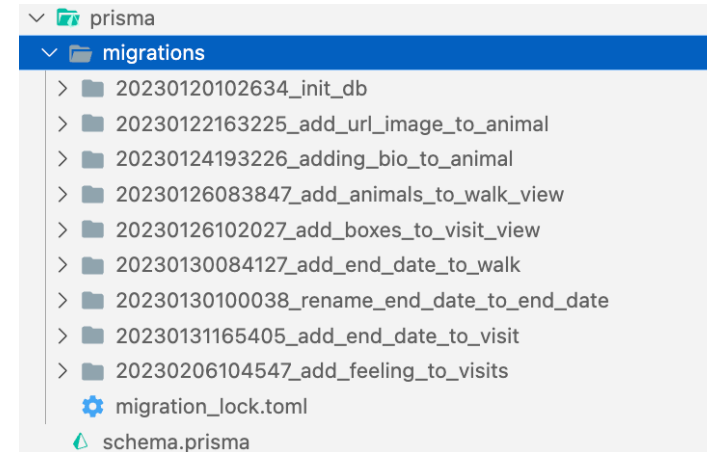
Architecture



Création de la base de données

```
9  datasource db {
10    provider = "postgresql"
11    url       = env("DATABASE_URL")
12  }
13
14  model User {
15    id          Int      @id() @default(autoincrement())
16    email       String   @unique()
17    password    String
18    phone_number String?
19    name        String
20    firstname   String
21    url_image   String?
22    admin       Boolean @default(false)
23    experience  String @default("BEGINNER")
24
25    walks Walk[]
26    visits Visit[]
27  }
```

Extrait du fichier *schema.prisma*



```
8  -- On vérifie que la valeur est égale à CAT, DOG ou OTHER
9  CREATE DOMAIN animal_specie AS TEXT
10 CHECK(
11   VALUE IN ('CAT','DOG','OTHER')
12 );
13
14 -- On vérifie le genre du chat
15 CREATE DOMAIN animal_gender AS TEXT
16 CHECK(
17   VALUE IN ('MALE','FEMALE')
18 );
```

Extrait d'un fichier *migration.sql*

A chaque modification du schema: **prisma migrate dev** et **prisma generate**

Accès aux données

- Pas de nécessité de créer des models
- Permet de réaliser des opérations CRUD

```
63  getAnimals: async (req, res, next) => {
64    try {
65      // on va chercher dans les animaux quels animaux ont l'id de la box
66      const animals = await prismaClient.animal.findMany({
67        where: {
68          box_id: Number(req.params.id),
69        },
70      });
71      res.json(animals);
```

```
136  const createdBox = await prismaClient.box.create({
137    data: {
138      type: req.body.type.toUpperCase() || 'OTHER',
139      nbr_of_places: Number(req.body.nbr_of_places),
140      number: req.body.number,
141    },
142  });
```

```
182  await prismaClient.box.delete({
183    where: {
184      id: Number(req.params.id),
185    },
186  });
```

Réalisation du back-end



Exemple: POST <https://api.toutopoils.fr/animals>

```
59 animalRouter.post(  
60     '/',  
61     authentication, // gestion de l'authentification  
62     fileUpload.single('image'), // gestion de l'upload de fichier, si présent  
63     validate(animalsValidation.create, 'body'), // validation des données du client  
64     animalsController.create  
65 );
```

Extrait du fichier /routes/animals.route.js


```

198 create: async (req, res, next) => {
199     if (req.user.admin === true) {
200         try {
201             /**
202              * ... vérification de l'existence de la box de l'animal
203              * ... gestion des tags ajoutés à l'animal
204              * ... gestion de l'upload d'image
205              */
206             const createAnimal = await prismaClient.animal.create({
207                 data: {
208                     species: animal.species || 'OTHER',
209                     name: animal.name,
210                     bio: animal.bio,
211                     gender: animal.gender,
212                     age: new Date(animal.age),
213                     size: animal.size,
214                     volunteer_experience: animal.volunteer_experience || 'BEGINNER',
215                     box_id: Number(animal.box_id),
216                     tags: {
217                         create: tagCreation,
218                     },
219                     url_image: animal.url_image,
220                 },
221             });
222
223             res.status(201).json(createAnimal);
224             //...

```

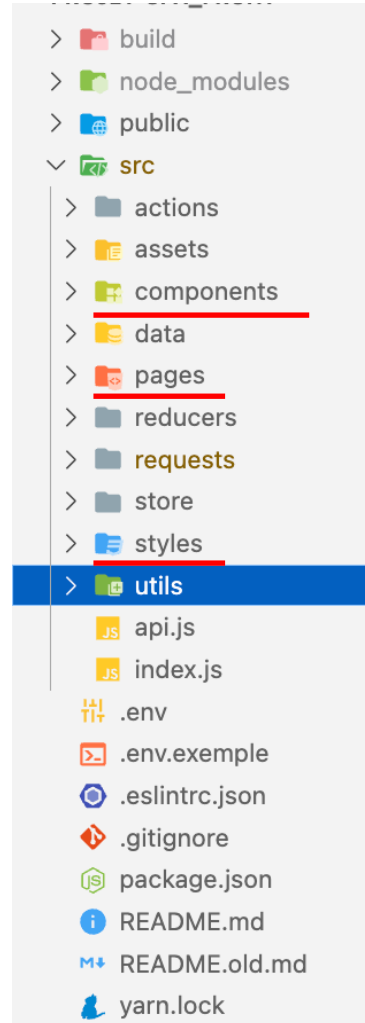
Extrait du fichier /src/controllers/animals.controller.js

Réalisation du front-end

partie « statique »

Application découpée en pages et composants

Router avec React-router



Structure du projet

```
20 const WalkSummary = (props) => {
21   const inProgress =
22     DateTime.fromISO(props.walk.date).plus({ hour: 1 }) >= DateTime.now() &&
23     !props.walk.end_date;
24
25   if (props.walk) {
26     return (
27       <div className={`animal-walks ${inProgress && 'walk-in-progress'}}>
28         <p>
29           le{' '}
30           {DateTime.fromISO(props.walk.date).toLocaleString(
31             DateTime.DATE_SHORT
32           )}{' '}
33         </p>
34         {DateTime.fromISO(props.walk.date).plus({ hour: 1 }) >=
35           DateTime.now() && !props.walk.end_date ? (
36           <div className='d-flex flex-row '
37             <div className='ms-auto '
38               <span className='tag-mood tag-mood--inprogress'>
39                 Balade en cours
40               </span>
41             </div>
42           </div>
43         ) : (
44           <div className='d-flex flex-row '
45             <p className='p-1'> {props.walk.comment} </p>
46             <div className='ms-auto p-1'
47               <WalkFeeling feeling={props.walk.feeling} />
48             </div>
49           </div>
50         )}
51       </div>
52     );
53   } else {
54     return null;
55   }
56 };
57
58 WalkSummary.propTypes = {
59   walk: PropTypes.shape({
60     id: PropTypes.number.isRequired,
61     user_id: PropTypes.number.isRequired,
62     date: PropTypes.string.isRequired,
63     comment: PropTypes.string,
64     feeling: PropTypes.string.isRequired,
65     end_date: PropTypes.string,
66   }),
67 };
```

Extrait de composant

Réalisation du front-end

Style

```
22 function Dashboard() {
23
24   // Récupération du nom de l'utilisateur dans le state
25   const firstName = useSelector(
26     (fullstate) => fullstate.loginSettings.firstName
27   );
28
29   return (
30     <>
31       <h1 className='title-page'>Bonjour {firstName} </h1>
32       <div className='dashboard-container'>
33         <Row>
34
35           <CardGroup>
36             <Col xs={12} md={6} lg={3}>
37               <Card className="dashboard-card" >
38                 <Card.Img variant='top' src={DogAndCat} />
39                 <Card.Body>
40                   <Button href='/animals' variant='primary' type='submit'>
41                     Voir tous les animaux
42                   </Button>
43                 </Card.Body>
44             </Card>
45           </Col>
```

Extrait de la page Dashboard

```
5 // Spacing
6 $gutter: 0.8rem;
7
8 // Colors variables
9 $primary-background-color: #fffaf7;
10 $secondary-background-color: #fdf3ed;
11 $primary-color: #eb651c;
12 $primary-color-hover: #35450a;
13 $font-color: #222;
14 $icons-primary-color: #928161;
15 $icons-secondary-color: #35450a;
16 $icons-background-color: #d9d9d9;
17
18 // Fonts
19 $title-font: "Amatic SC", cursive;
20 $text-font: "Roboto", sans-serif;
21
22 // Fonts size
23 $h1-font-size: 3.5rem;
24 $h2-font-size: 2rem;
25 $h3-font-size: 1.5rem;
26 $font-size: 1rem;
27 $subtitle: 1.5rem;
28
29 // Font weight
30 $title-weight: 700;
31
32
33 // Shadow
34 $shadow: 1px 1px 36px rgba(0, 0, 0, 0.09);
```

Extrait du fichier /styles/_vars.scss

Front

partie « dynamique »

- **Gestion du state**
 - Redux
 - Redux Toolkit
- **Requêtes**
 - Axios
 - React Query
- **Gestion des formulaires**
 - React hook forms
 - Yup (validation)

```
72 get: (id, options) => {
73     // Création de la liste des includes
74     let includes = [];
75     options?.includeTags && includes.push('tags');
76     options?.includeWalks && includes.push('walks');
77     options?.includeBox && includes.push('box');
78     // Création des objets à inclure dans la query
79     let queryBuilder = {
80         include: includes,
81     };
82     // Conversion de l'objet de query en string à passer dans la requête
83     const query = qs.stringify(queryBuilder, {
84         skipNulls: true,
85         arrayFormat: 'comma',
86     });
87
88     // retourne un requête du style : /animals/1?include=tags,walks
89     // => récupère l'animal 1 avec ses tags et ses balades
90     return api.get(`/animals/${id}?${query}`);
91
92 },
```

Exemple de requête

Sécurité

FAIT	A FAIRE
<ul style="list-style-type: none">- Validation des données<ul style="list-style-type: none">- Yup (front) & Joi (back)- Mot de passe hashés- Utilisation d'un captcha en front- XSS: React sécurisé « par défaut »- Prisma: éviter les injections SQL	<ul style="list-style-type: none">- Rate limiting au niveau de l'API- Meilleure configuration du CORS

RÉALISATIONS PERSONNELLES



Front

Mise en place des requêtes vers le back

- base: instance axios
- Requetes groupées dans des objets
- Utilisation de React Query

Amélioration possible

- Utiliser React Query de façon plus efficace

```
1 import axios from 'axios';
2
3 // récupération du token dans le local storage
4 export const getToken = () =>
5   localStorage.getItem('token')
6     ? JSON.parse(localStorage.getItem('token'))
7     : null;
8
9 // création d'une instance d'axios
10 const api = axios.create({
11   baseURL: `${process.env.REACT_APP_API_URL}/v1`,
12 });
13
14 // Middleware axios qui permet de conserver le token lors de chaque requête au back
15 api.interceptors.request.use((config) => {
16   config.headers['Authorization'] = `Bearer ${getToken()}`;
17   return config;
18 });
19
20 export default api;
```

```
8  getProfile: (id) => {
9    return api.get(`/users/${id}`);
10 },
```

```
160 const { isLoading: loading, error } = useQuery(
161   'LoadProfile',
162   () => usersRequest.getProfile(id),
163   {
164     onSuccess: (data) => {
165       if (data.data) {
166         setUserData(data.data);
167       }
168     },
169   }
170 );
```

Back

Controller de récupération du JWT

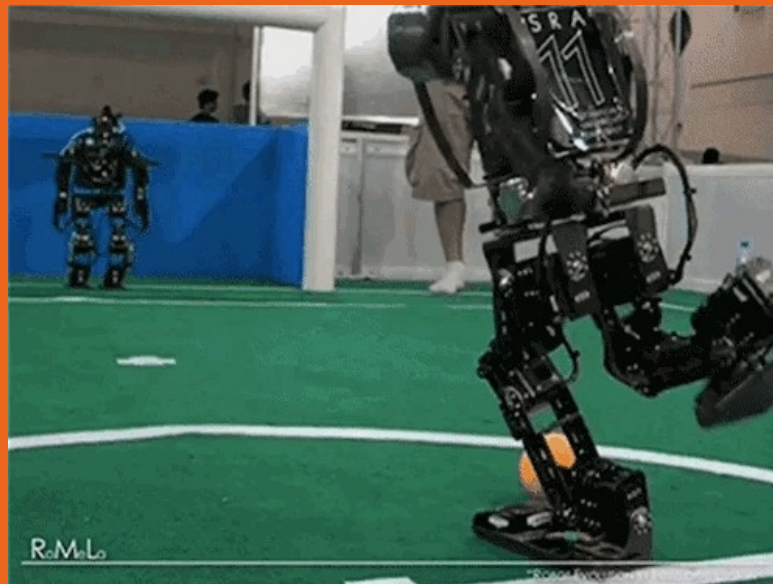
- Mot de passe hashés avec bcrypt
- Comparaison du mot de passe
- Génération d'un JWT
- Séparation des méthodes dans un service dédié

Amélioration possible:

- En cas d'erreur, une seule et unique erreur, par exemple « INVALID_CREDENTIALS »

```
7  const authController = {
8    login: async (req, res, next) => {
9      const { email, password } = req.body;
10     try {
11       // on vérifie que l'utilisateur existe avec cet email
12       const user = await prismaClient.user.findUnique({
13         where: {
14           email,
15         },
16       });
17
18       if (user) {
19         // on vérifie que le mot de passe est correct
20         const passwordValidation = await authService.validatePassword(
21           password,
22           user.password
23         );
24
25         if (passwordValidation) {
26           // on génère un JWT signé avec les informations de l'utilisateur
27           const signedJwt = await authService.generateJWT({
28             id: user.id,
29             admin: user.admin ?? false,
30             firstName: user.firstname,
31             experience: user.experience ?? 'BEGINNER',
32             url_image: user.url_image ?? null,
33           });
34           // on retourne un JWT signé pour vérifier sa validité
35           res.json({ token: signedJwt });
36         } else {
37           res.status(401).json({ message: 'INVALID_PASSWORD' });
38         }
39       } else {
40         res.status(401).json({ message: 'INVALID_USER' });
41       }
42     }
43     //...
```


DÉMO DU PROJET





Application de gestion des actions de bénévoles



Email



Mot de passe



I'm not a robot



reCAPTCHA
Privacy - Terms

SE CONNECTER

EXEMPLE DE RECHERCHE



Présentation d'une recherche


- Recherche en anglais, avec des mots simples
- Sélection des sources:
 - Classement « personnel »
 - Vérification de la date de création ou de mise à jour

The screenshot shows a Google search interface. The search bar contains the text "prisma vs raw sql". Below the search bar, there are tabs for "Tous", "Images", "Vidéos", "Shopping", "Actualités", and "Plus". The search results show "Environ 2 400 000 résultats (0,23 secondes)". The first result is from Prisma.io, titled "Why Prisma? Comparison with SQL query builders & ORMs". The second result is from Stack Overflow, titled "prisma - Orm or RAW sql which one is better?".

prisma vs raw sql


Tous Images Vidéos Shopping Actualités Plus Outils

Environ 2 400 000 résultats (0,23 secondes)

 Prisma.io
<https://www.prisma.io> › overview · [Traduire cette page](#)

Why Prisma? Comparison with SQL query builders & ORMs

On this page, you'll learn about the motivation for **Prisma** and how it compares to other database tools like ORMs and **SQL** query builders.

 Stack Overflow
<https://stackoverflow.com> › questions · [Traduire cette page](#)

prisma - Orm or RAW sql which one is better?

4 mai 2022 · 1 réponse

ORM and RAW SQL both have their own pros and cons. But if you're looking for a simple and easy way to use a database, ORM is the way to go.

Présentation d'une recherche


Classement personnel:

1. Documentation
2. Un acteur ou un groupe d'acteurs connus
 - ex: OWASP
3. Sites ayant attiré à la programmation
 - ex: blox LogRocket
4. Stackoverflow
 - questions théoriques
 - éviter le code

prisma vs raw sql


Tous Images Vidéos Shopping Actualités Plus Outils

Environ 2 400 000 résultats (0,23 secondes)

 Prisma.io
<https://www.prisma.io> › overview · Traduire cette page

Why Prisma? Comparison with SQL query builders & ORMs

On this page, you'll learn about the motivation for **Prisma** and how it compares to other database tools like ORMs and **SQL** query builders.

 Stack Overflow
<https://stackoverflow.com> › questions · Traduire cette page

prisma - Orm or RAW sql which one is better?

4 mai 2022 · 1 réponse

ORM and Raw SQL both have their own pros and cons. But it's not like to use model depend on...

Présentation d'une recherche

Dans cet exemple

- Doc de Prisma
- Pourquoi utiliser Prisma
- Point de vigilance: sujet à discussion

Concepts / Overview

Why Prisma?

On this page, you'll learn about the motivation for Prisma and how it compares to other database tools like ORMs and SQL query builders.

Working with relational databases is a major bottleneck in application development. Debugging SQL queries or complex ORM objects often consume hours of development time.

Prisma makes it easy for developers to reason about their database queries by providing a clean and type-safe API for submitting database queries which returns *plain old JavaScript objects*.

TLDR

Prisma's main goal is to make application developers more productive when working with databases. Here are a few examples of how Prisma achieves this:

ON T

TLDR

Prob
and c

Appli
shou
not S

Prism
prod

Présentation d'une recherche

Raw SQL: Full control, low productivity

With raw SQL (e.g. using the native `pg` or `mysql` Node.js database drivers) you have full control over your database operations. However, productivity suffers as sending plain SQL strings to the database is cumbersome and comes with a lot of overhead (manual connection handling, repetitive boilerplate, ...).

Another major issue with this approach is that you don't get any type safety for your query results. Of course, you can type the results manually but this is a huge amount of work and requires major refactorings each time you change your database schema or queries to keep the typings in sync.

Furthermore, submitting SQL queries as plain strings means you don't get any autocompletion in your editors.

SQL query builders: High control, medium productivity

A common solution that retains a high level of control and provides better productivity is to use a SQL query builder (e.g. [knex.js](#)). These sort of tools provide a programmatic abstraction to construct SQL queries.

The biggest drawback with SQL query builders is that application developers still need to think about their data in terms of SQL. This incurs a cognitive and practical cost of translating relational data into objects. Another issue is that it's too easy to shoot yourself in the foot if you don't know exactly what you're doing in your SQL queries.

Conclusion

- Satisfaction du travail accompli
- Bonne équipe
- Appréciation du travail en équipe
- Réflexions sur la poursuite du projet

A vos questions !