Projet Fil Rouge

CDA34 24-01

Botta Facundo

Sommaire

About me...

Facundo Botta

Portfolio

GitHub

LinkedIn

Dev community

The project: **EventHub**

What about?

- All-in-one platform for events & appointment management
- For professionals & regular users
- Simplifies organization & communication

Why this app?

- Options in the market
- First approach -> Appointment management system
- Finally -> More like a social network

CDC - Analyse du besoin

Fonctions du produit

- Gérer commentaires, ponctuation et signalement des utilisateurs
- Authentification avec compte Google et intégration avec son calendrier
- Administrer les événements, notifier les utilisateurs
- Système de messagerie et liste de tâches
- Système de recherche (utilisateurs, événements, messages)

CDC - Analyse du besoin

Contraintes techniques

- Base de données large et complexe
- Sécurité du site et protection des données
- Integration sur mobile

CDC - SWOT

S
Strengths
FORCES

- Faible concurrence
- Intuitif
- Gratuit

W
Weaknesses
FAIBLESSES

- Peu connu
- Confiance
- Limitations

O
Opportunities
OPPORTUNITES

- Innovation
- Evolution

T
Theats
MENACES

- Conflits entre utilisateurs

CDC - Public visé

Nom

"Claude Martin"

- Age : 34
- Travail : Personal Trainer
- Ville : Montpellier

Bio

Il propose ses services de formation personnelle pour les groupes et les particuliers. L'application le permet de gérer efficacement son emploi du temps, d'accepter de nouveaux rendez-vous et de conserver un enregistrement organisé de ses clients et de ses sessions.

Personnalité

Introverti — Extrovert

Occupé — Créatif

Analytique — Temps libre

Déssordonné — Organisé

Indépendant — Esprit d'équipe

Nom

"Jean Pierre"

- Age : 68
- Travail : À la retraite
- Ville : Montpellier

Bio

Bien que lui puisse bénéficier des services professionnels disponibles dans l'applapplication , il recherche spécifiquement un chirurgien certifié. Pour ses besoins, un site comme Doctolib serait plus approprié.

Personnalité

Introverti — Extrovert

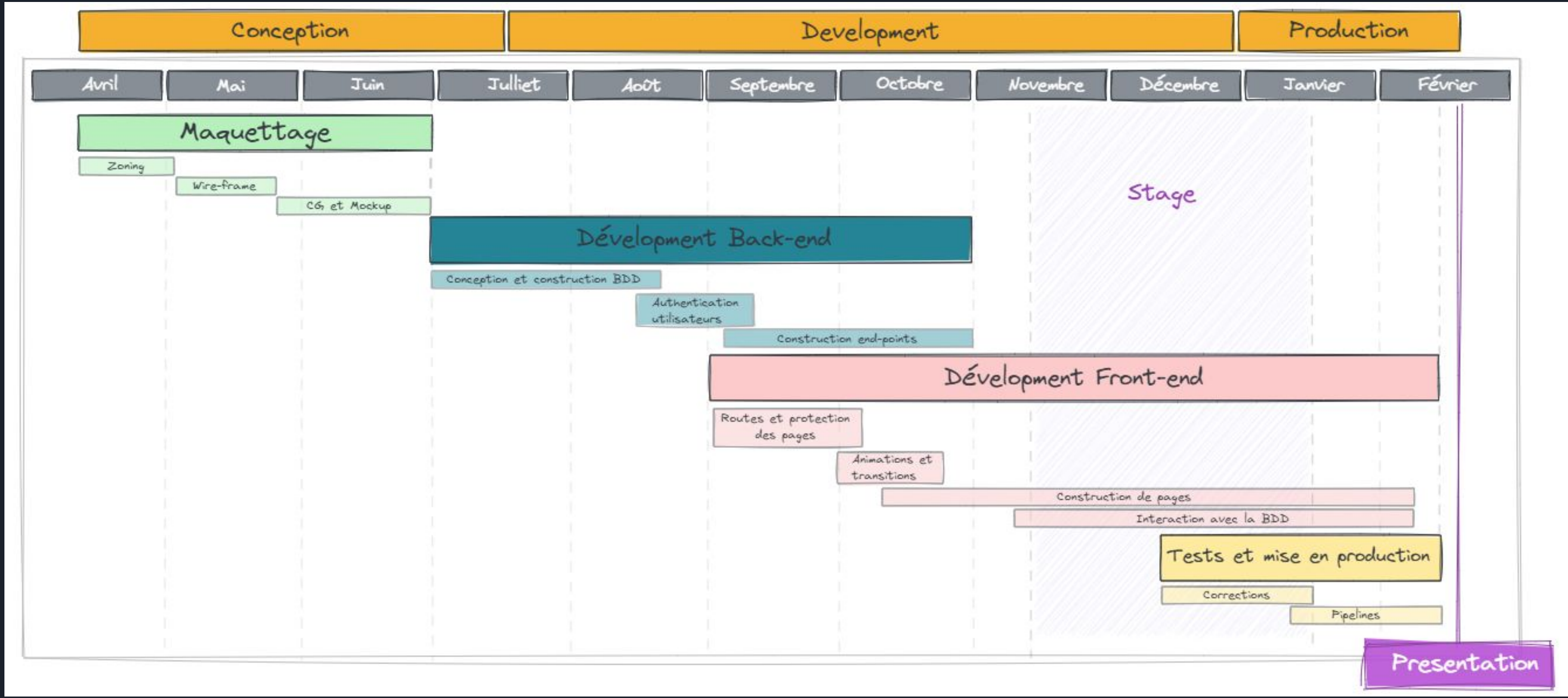
Occupé — Créatif

Analytique — Temps libre

Déssordonné — Organisé

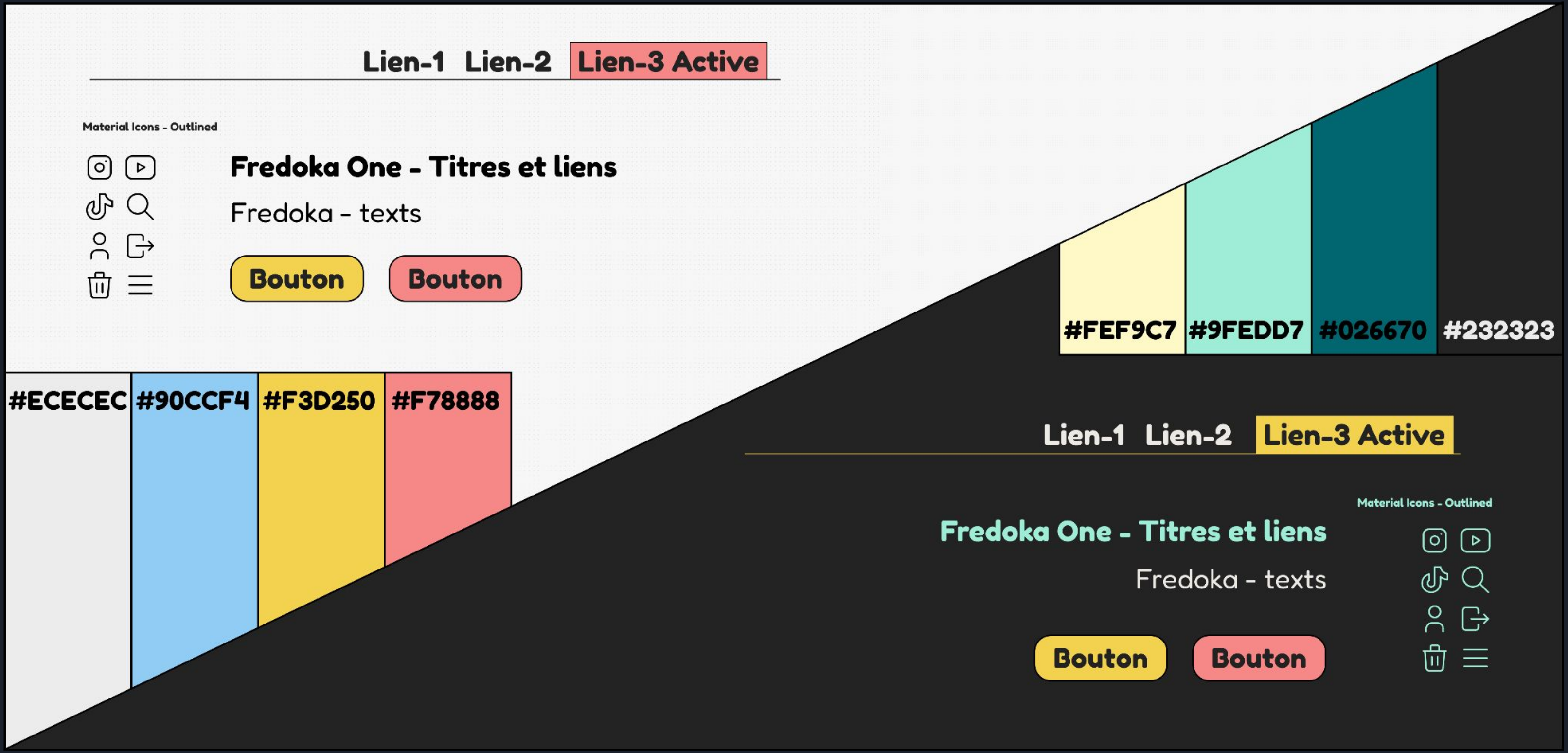
Indépendant — Esprit d'équipe

CDC - Gantt



Conception
Development
Production
Avril
Mai
Juin
Julliet
Août
Septembre
Octobre
Novembre
Décembre
Janvier
Février
Maquettage
Zoning
Wire-frame
CG et Mockup
Stage
Dévelopment Back-end
Conception et construction BDD
Authentication utilisateurs
Construction end-points
Dévelopment Front-end
Routes et protection des pages
Animations et transitions
Construction de pages
Interaction avec la BDD
Tests et mise en production
Corrections
Pipelines
Presentation

CDC - Style tile



CDC - Sélection des technologies

JavaScript

CSS3

HTML5

MySQL

TypeScript

Tailwind CSS

React

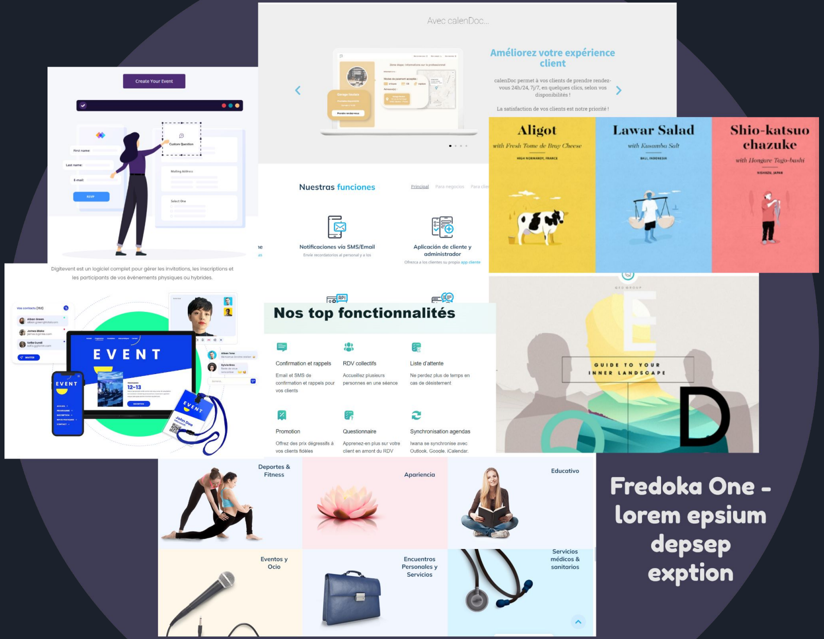
Prisma

Zod

Next.js

Auth.js

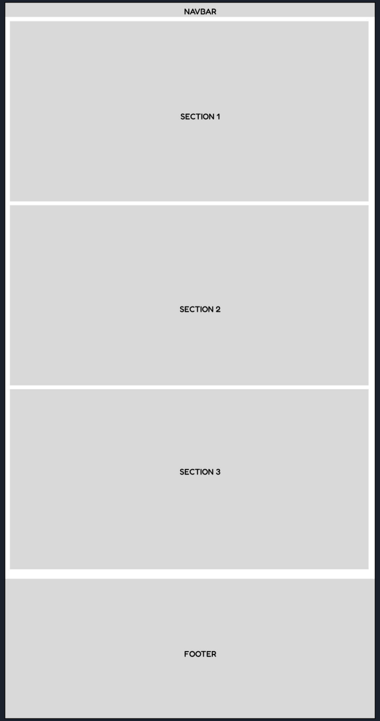
Maquettage - MoodBoard



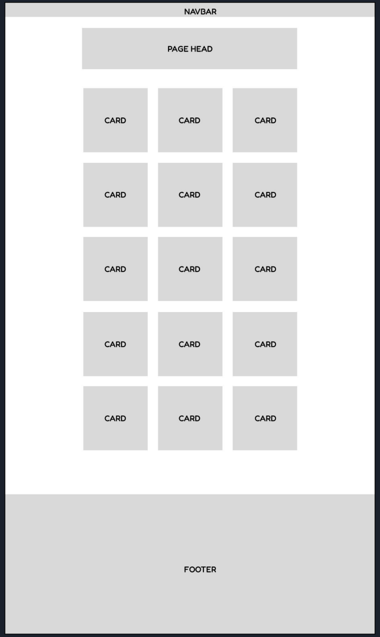
Fredoka One - lorem epsium depsep exption

Maquettage - Zoning

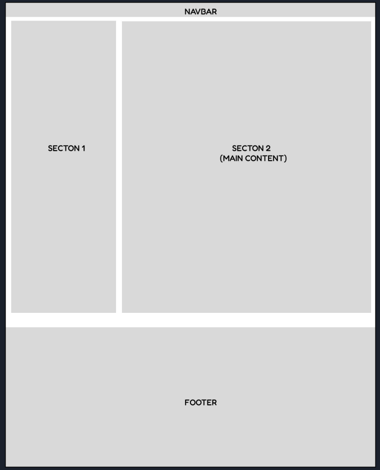
Accueil



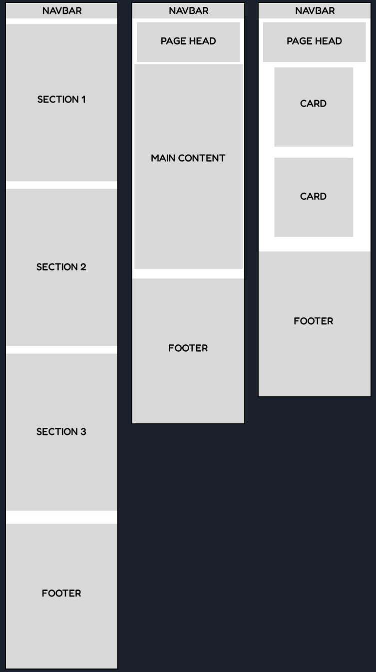
Communauté /Événements



Messagerie/Profil

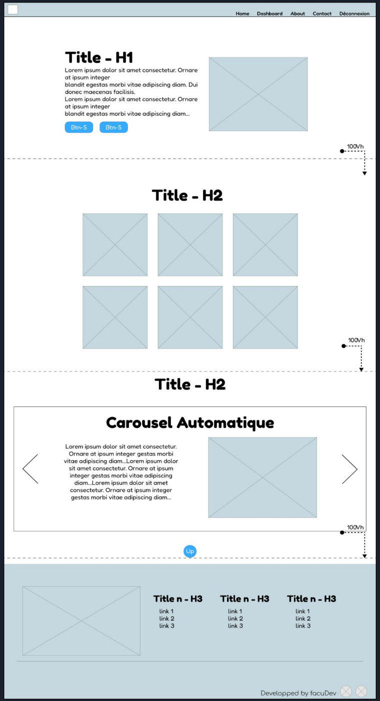


Petits écrans

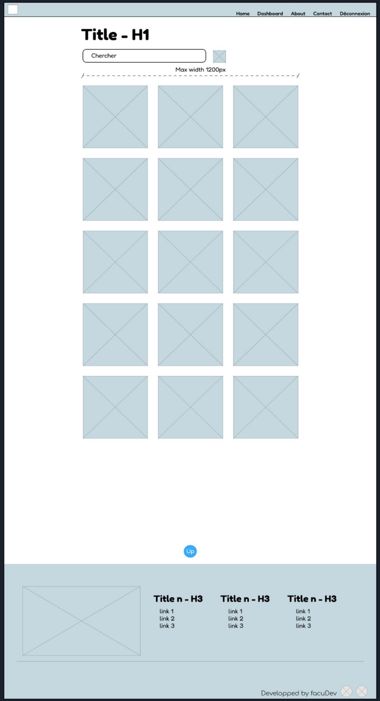


Maquettage - WireFrame

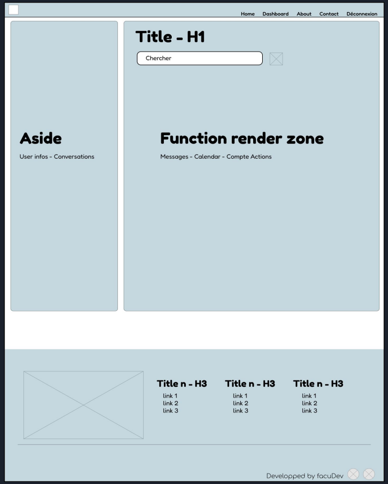
Accueil



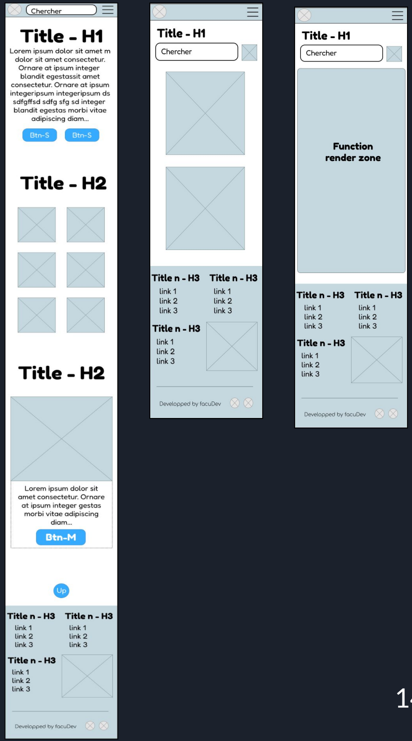
Communauté/Événements



Messagerie/Profil

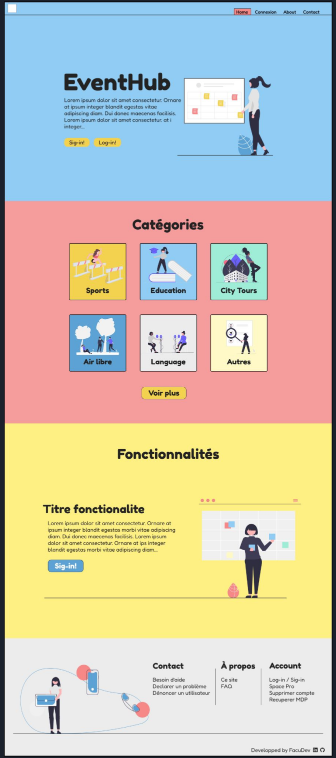


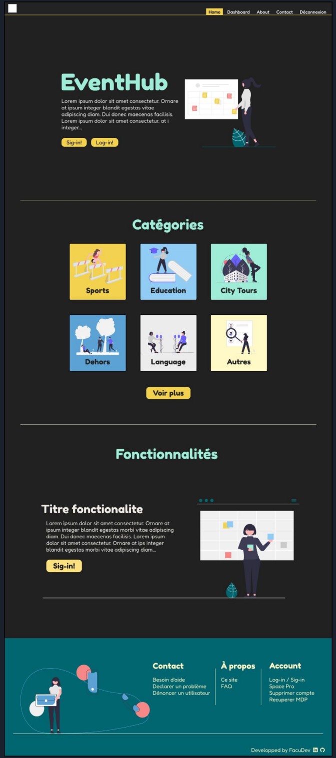
Petits écrans



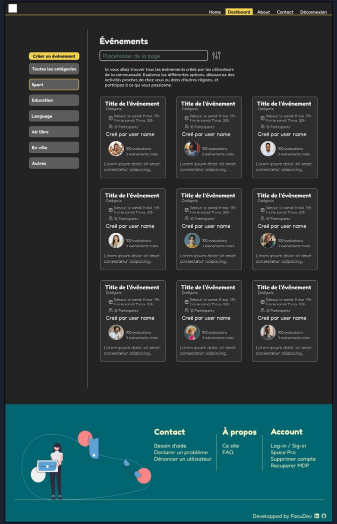
Maquettage - Mockup - 1440px

Accueil

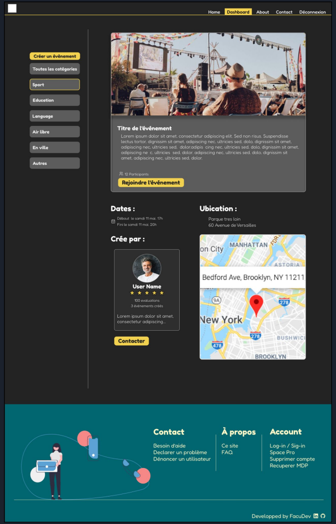




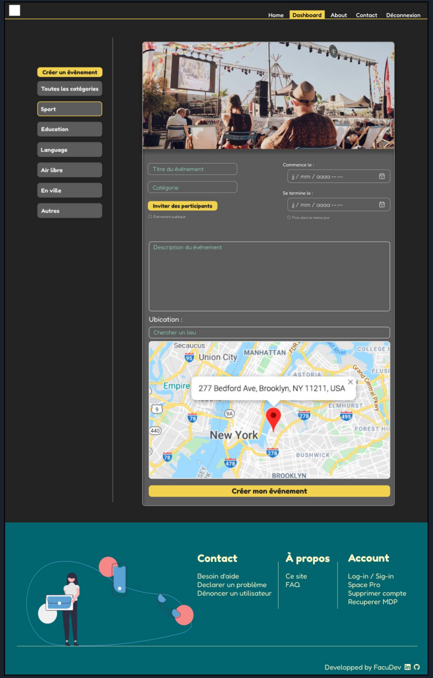
Événements



Événement/:id



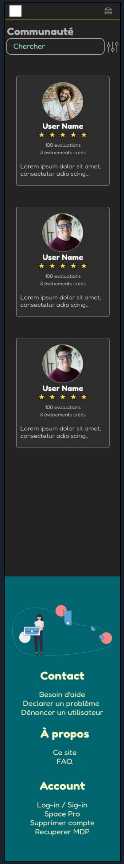
Événement/création



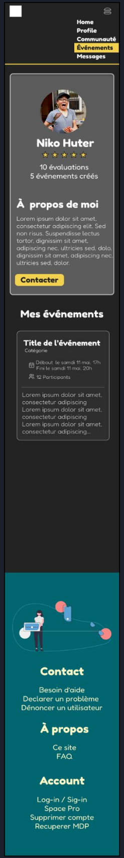
Maquettage - Mockup - 360px

ACCUEIL

COMMUNAUTÉ

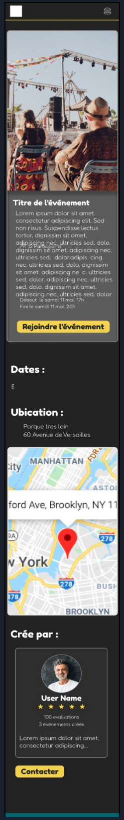


PROFIL/:ID

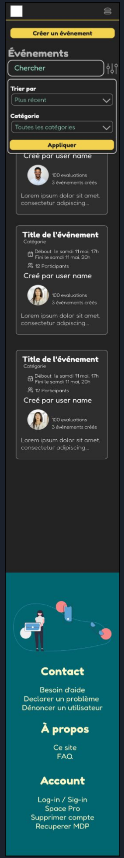


MON COMPTE

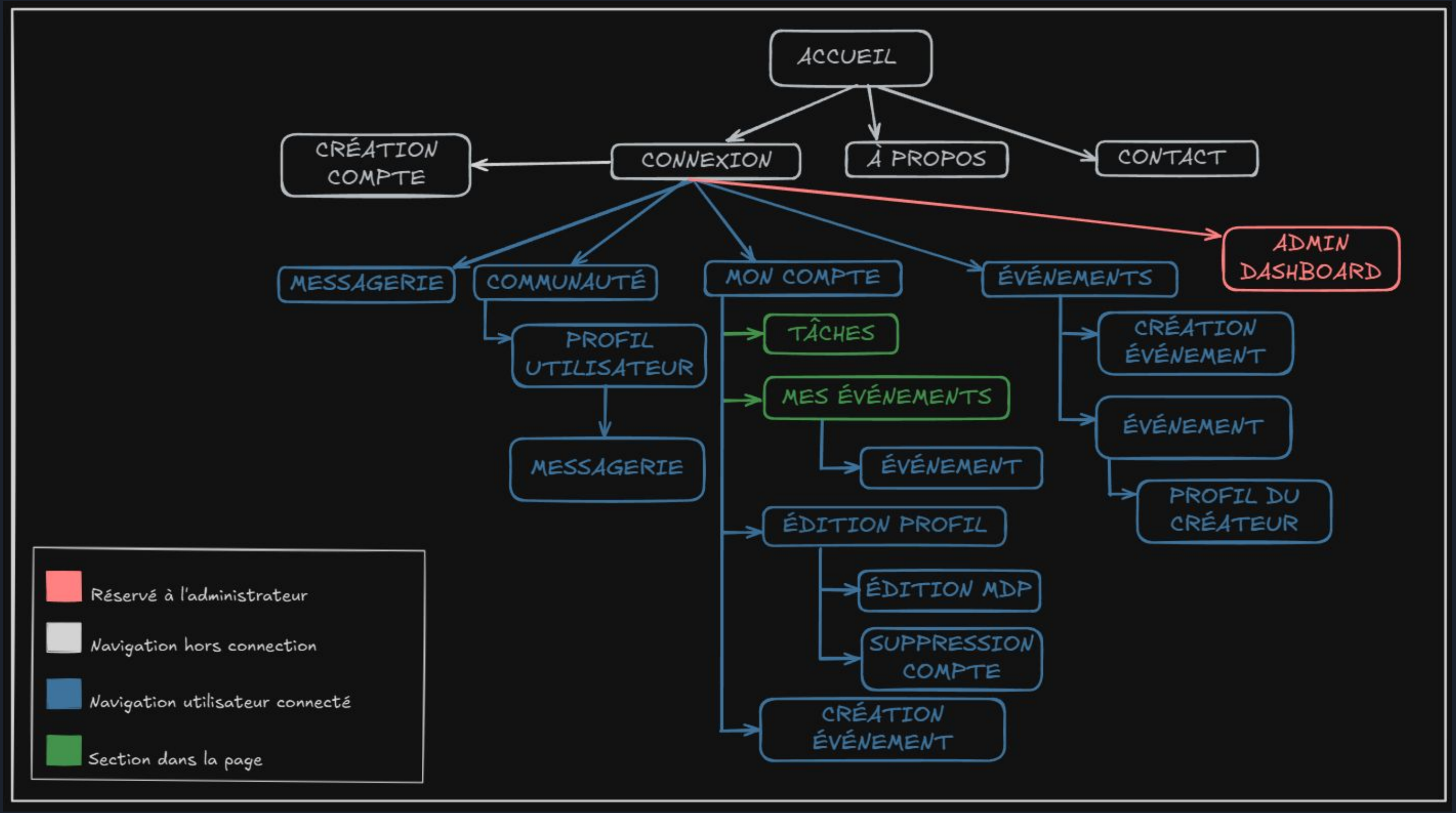
ÉVÉNEMENT



ÉVÉNEMENTS



Conception - Arborescence du site

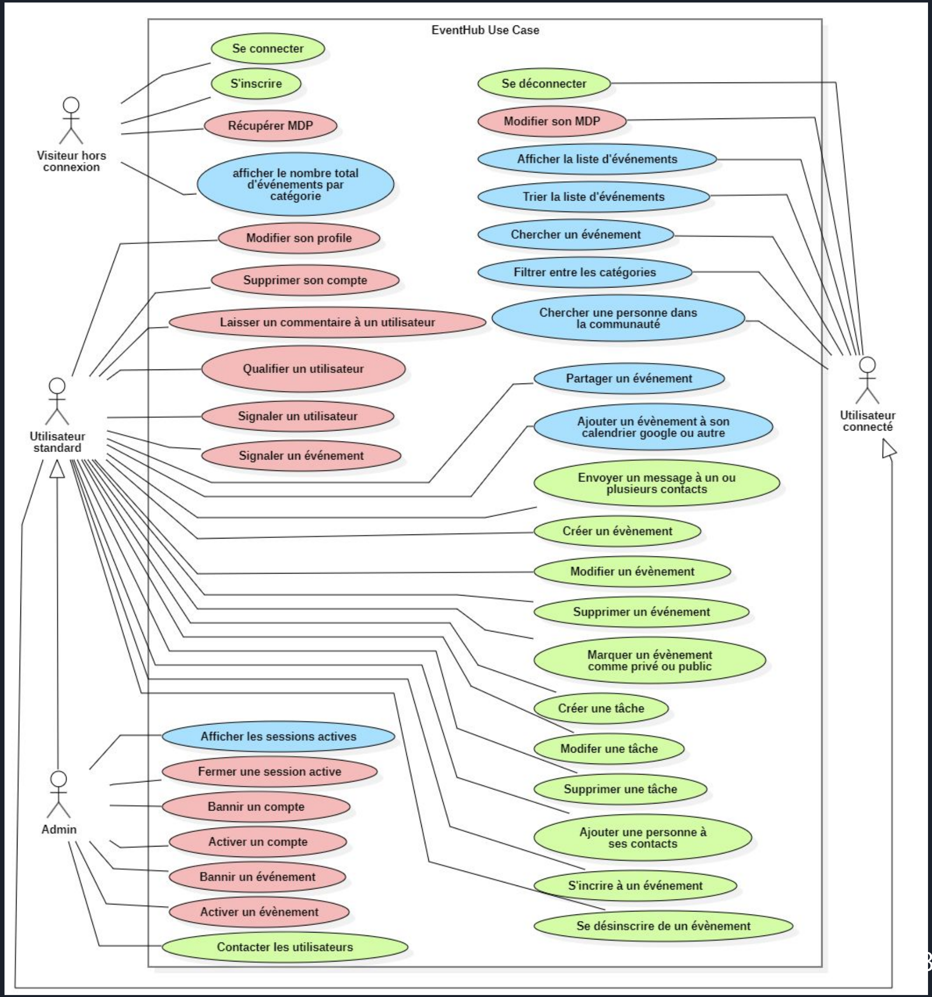


Conception - UML - Use Case

Actions ayant un impact sur les comptes utilisateurs.

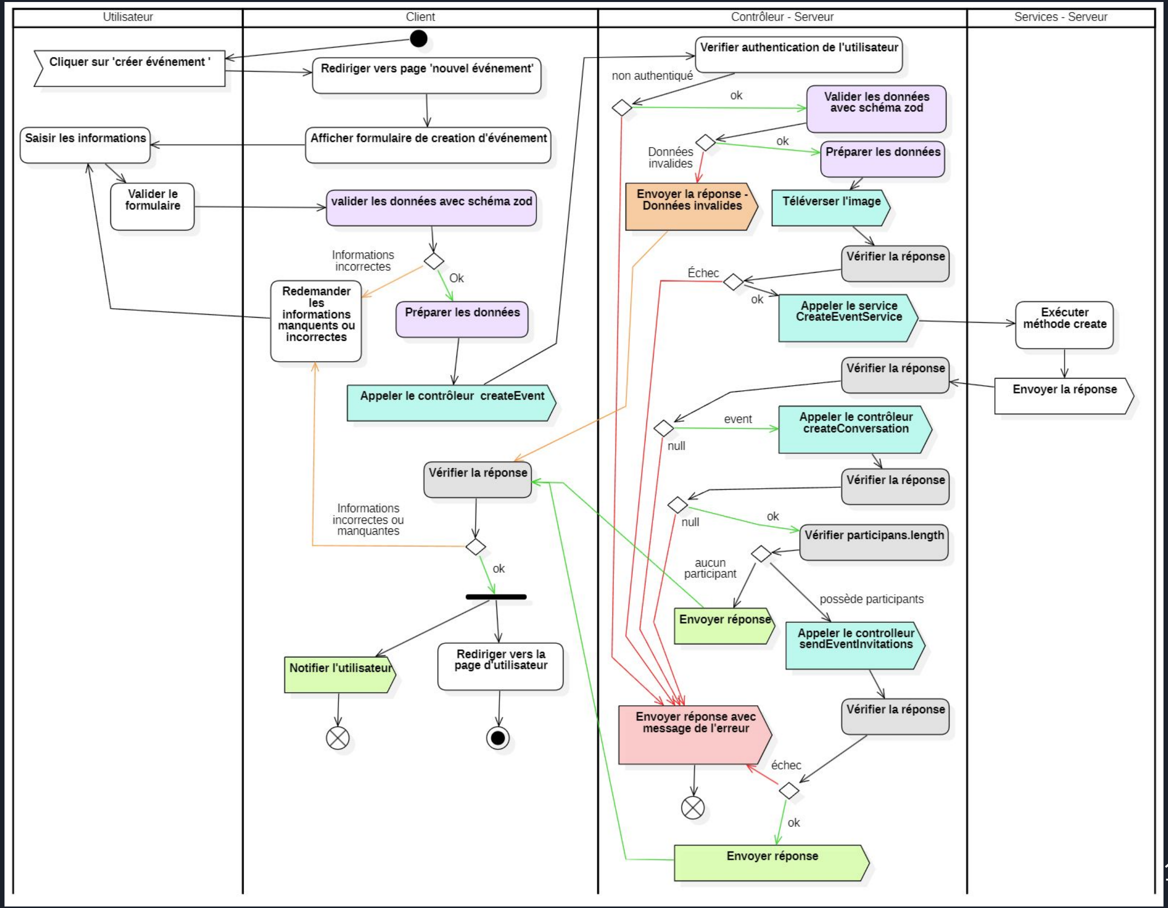
Actions sans création d'éléments dans la base de données.

Actions avec un impact sur la base de données.

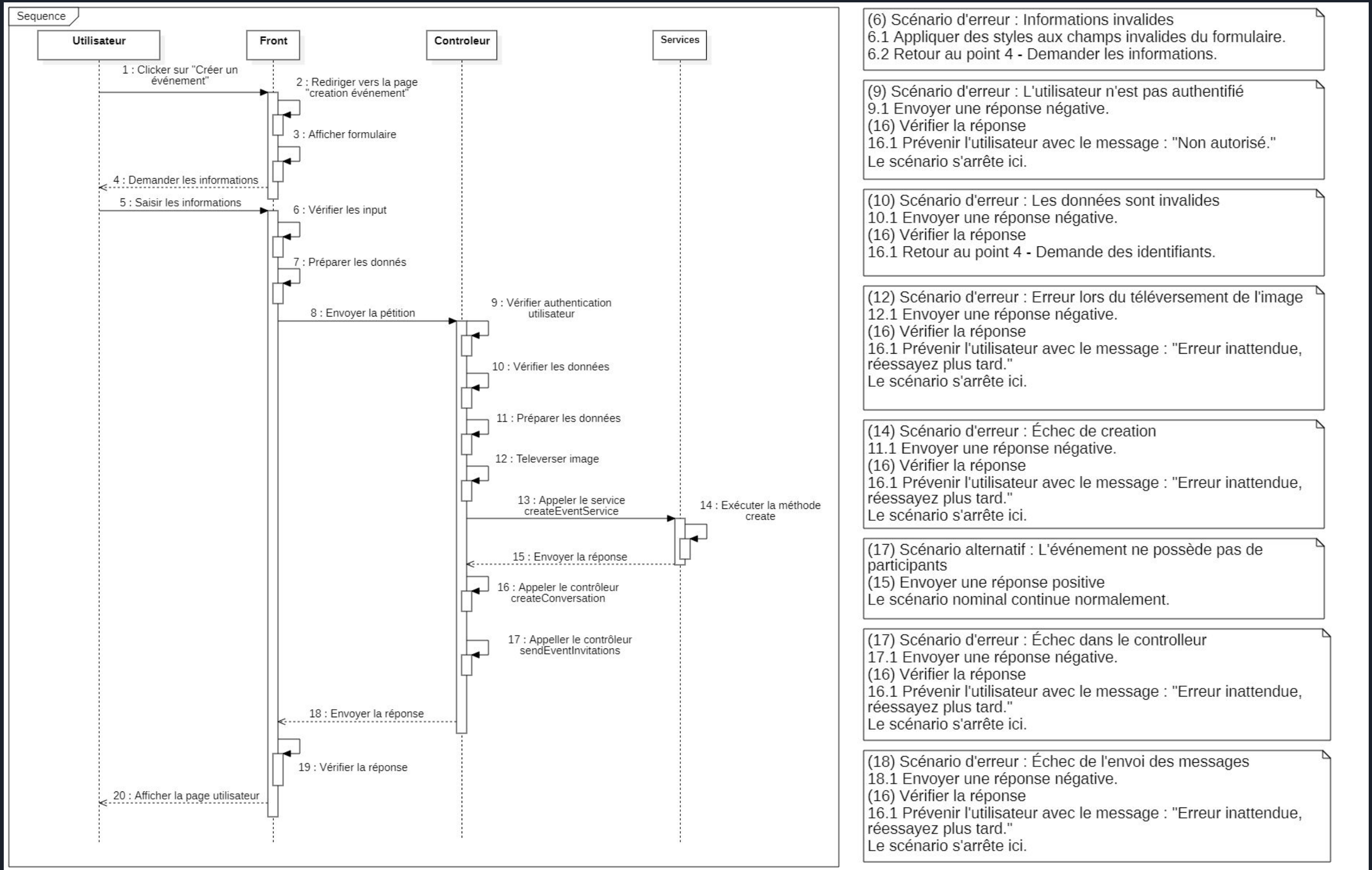


Conception - UML - Activité - Création événement

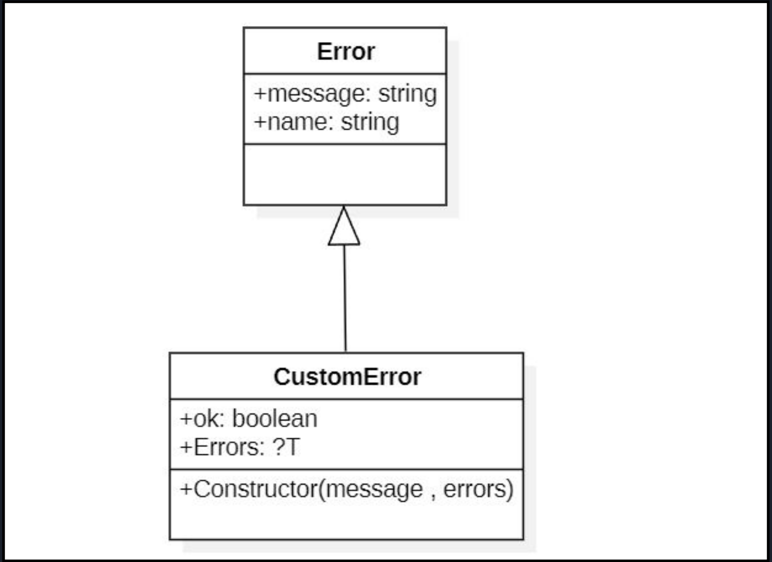
- Appels asynchrones
- Manipulation des données
- Validation des données
- Réponse négative sans arrêter le flow
- Réponse négative arrêtant le flow
- Réponse positive



Conception - UML - Séquence - Création événement

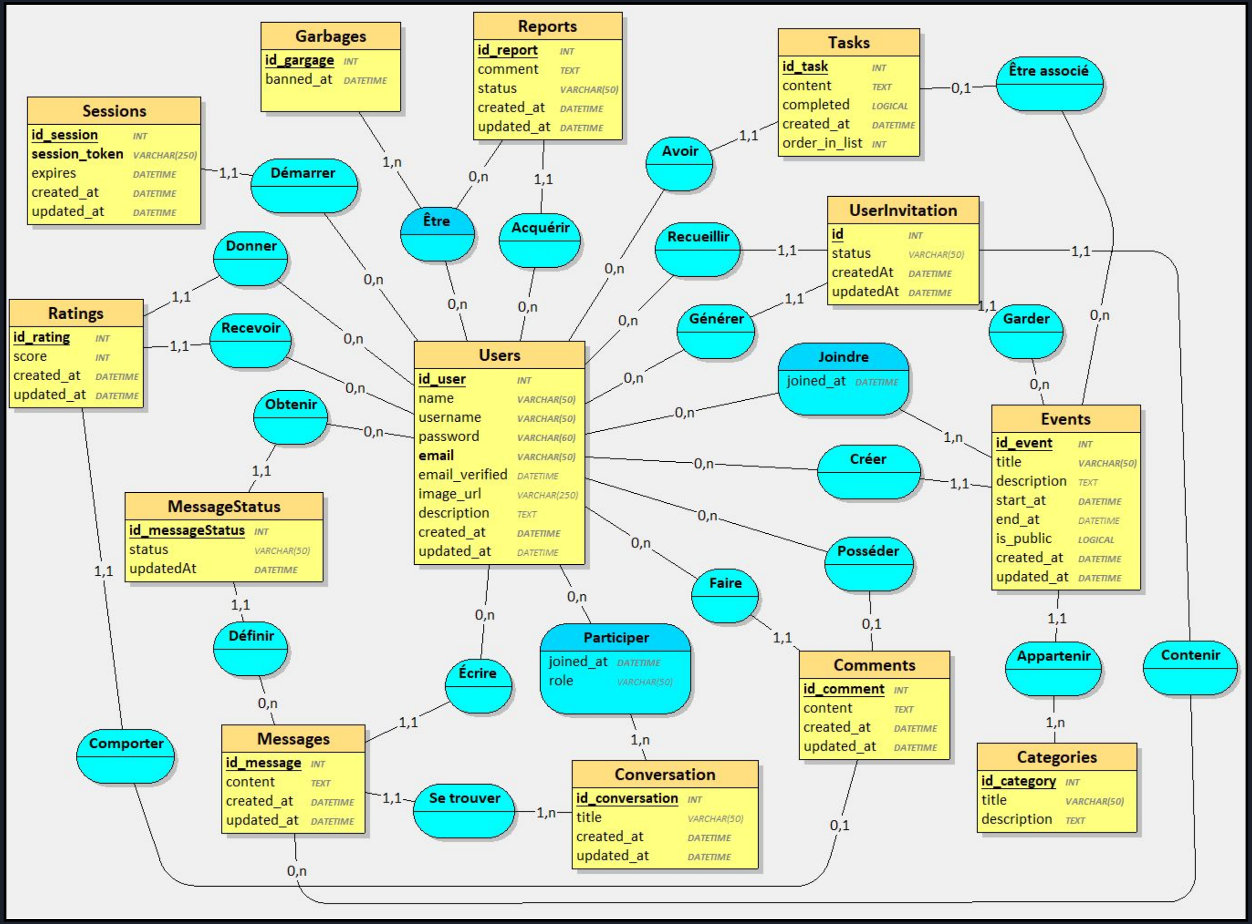


Conception - UML - Classe - CustomError



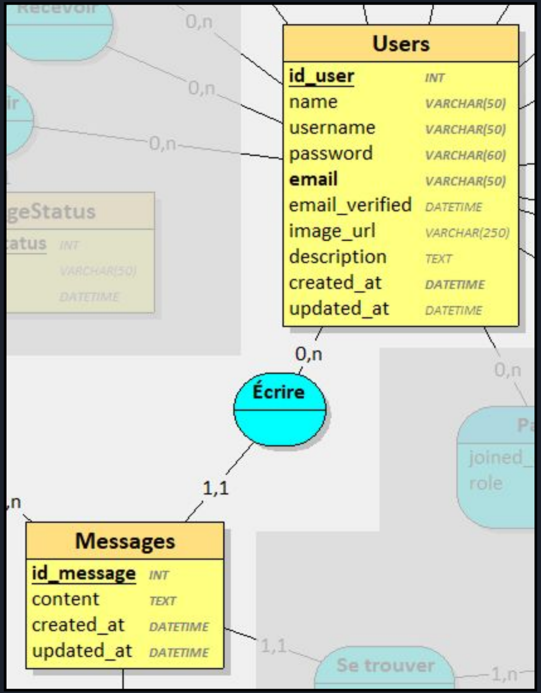

```typescript
/**
 * Represents a custom error that extends the built-in `Error` class.
 * This class allows for additional structured error information.
 *
 * @template T - The type of additional error details (e.g., validation errors).
 */
class CustomError<T = unknown> extends Error {
  /**
   * Indicates that the operation was not successful.
   * Always `false` for instances of `CustomError`.
   */
  public ok: boolean;

  /**
   * A error message describing the issue.
   */
  public message: string;

  /**
   * Additional error details, such as validation errors or metadata.
   * This field is optional.
   */
  public errors?: T;

  /**
   * Creates an instance of `CustomError`.
   *
   * @param {string} message - A descriptive error message.
   * @param {T} [errors] - Optional additional error details.
   */
  constructor(message: string, errors?: T) {
    super(message);
    this.name = this.constructor.name;
    this.ok = false;
    this.message = message;
    this.errors = errors;
  }
}

export default CustomError;
```

Conception - MCD

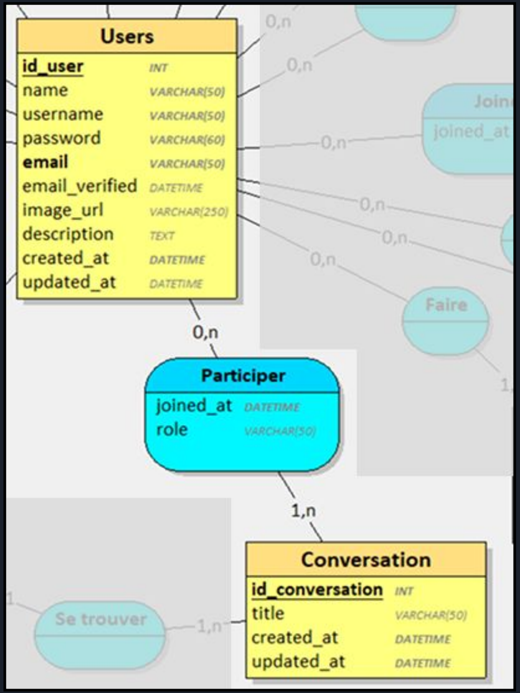


Conception - MCD

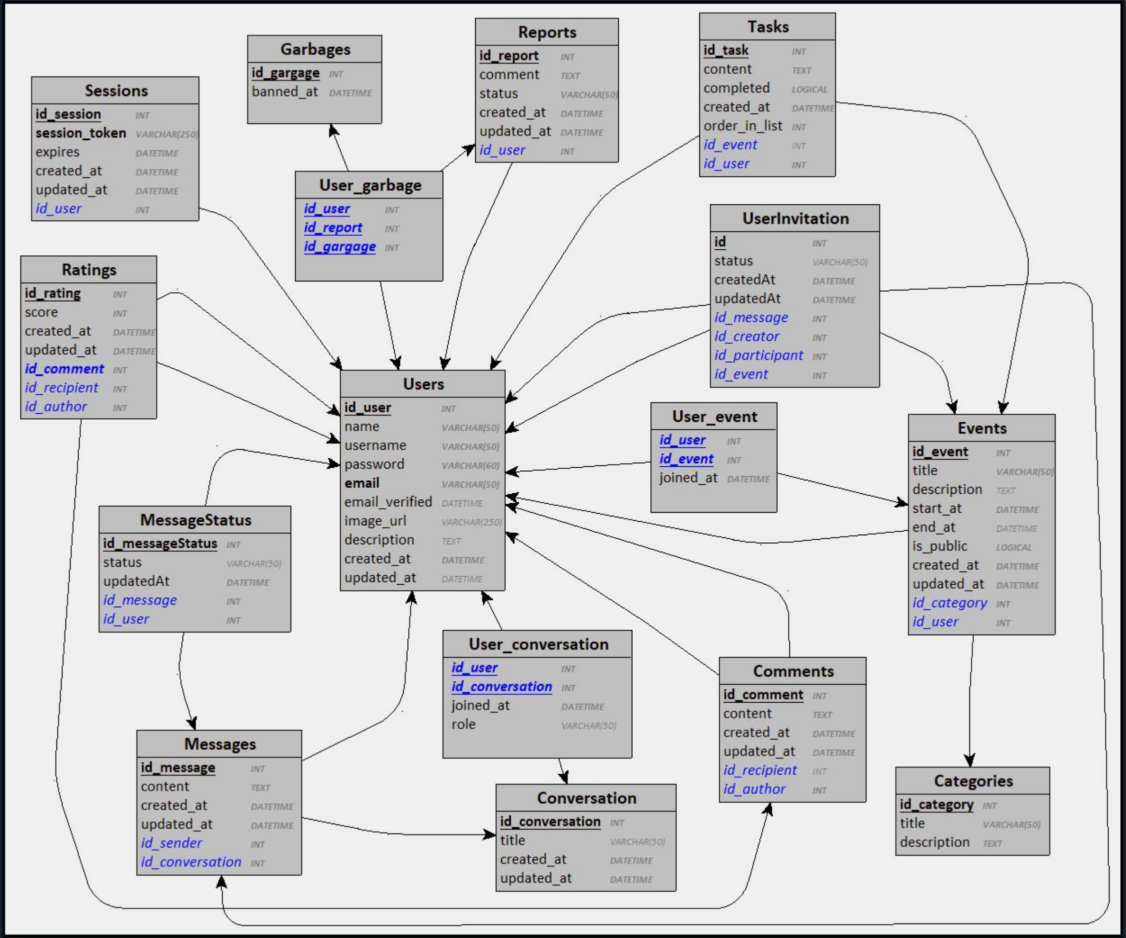
Contrainte d'Intégrité Fonctionnelle (CIF)



Contrainte d'Intégrité Multiple (CIM)

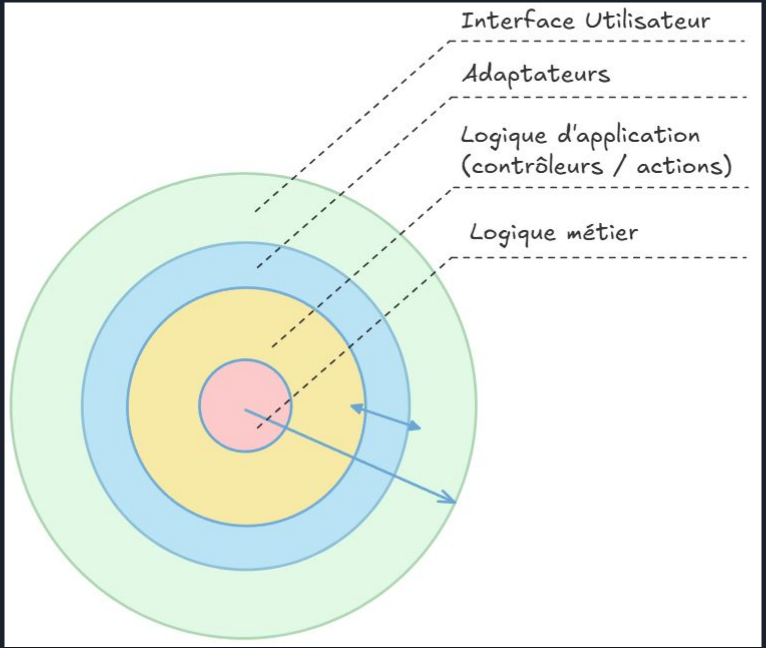


Conception - MLD

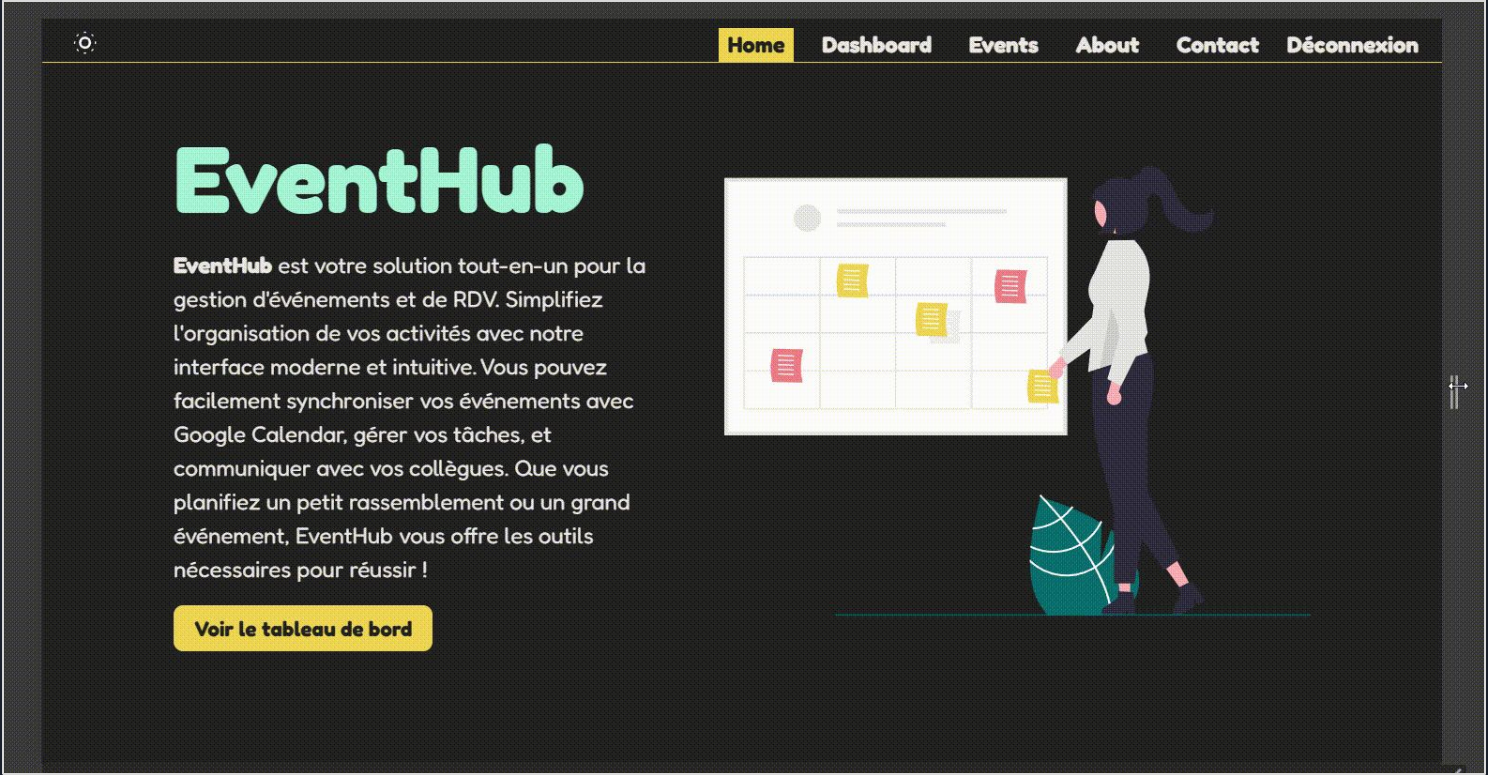


Architecture - Clear Architecture

```
/app
    /about
    /contact
    /login
    etc...
    layout.tsx
    not-found.tsx
    page.tsx
/components
    /ui
    /layout
    /forms
/actions
    /authServerActions
        checkIsAuthenticated.ts
        etc...
        authActions.test.ts
    /userServerActions
        selectUserById.ts
        etc...
        userActions.test.ts
/lib
    /prisma
        /migrations
        prisma.ts
        prisma_schema.prisma
    /auth
        authConfig.ts
        next-auth.d.ts
        emailTemplate.ts
    /zod
        zodSchemas.ts
        handleError.ts
        handleError.test.ts
/services
    eventService.ts
    event.test.ts
    etc...
/types
    appTypes.ts
    servicesTypes.ts
/utils
    adpapters/
        userAdapters.ts
        userAtapters.test.ts
        etc...
    formatDates.ts
    getFormDataValues.ts
    calculateUserScore.ts
    etc...
middleware.ts
tailwind.config.ts
tsconfig.json
package.json
next.config.ts
.env
```



Fonctionnalité Front - Menu de navigation avec 0 JavaScript



Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```jsx
export default async function Header() {

  return (
    <>
      <input type="checkbox" id="menu-toggle" className="hidden peer" />
      <header className="sticky z-50 top-0 min-w-full sm:max-h-none sm:items-end flex justify-between overflow-hidden
      max-h-[44px] transition-[max-height] duration-500 ease-in-out peer-checked:max-h-[500px] select-none px-5 border-b-
      [1px] border-dark-bg dark:border-light-yellow bg-light-ciel/95 dark:bg-dark-bg/95 ">
        <ThemeSwitcher />
        <nav>
          <div className="flex flex-col items-end">
            <label
              id="menu-label"
              htmlFor="menu-toggle"
              className="cursor-pointer sm:hidden p-2 active:scale-95 "
            >
              <NavMenuButton />
            </label>
            <ul className="mr-[-1.25rem] sm:mr-0 gap-4 font-bold text-dark-bg dark:text-dark-grey sm:flex sm:flex-row">
            </ul>
          </div>
        </nav>
      </header>
    </>
  );
}
```

Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```
export default async function Header() {

  return (
    <>
      <input type="checkbox" id="menu-toggle" className="hidden peer" />
      <header className="sticky z-50 top-0 min-w-full sm:max-h-none sm:items-end flex justify-between overflow-hidden
      max-h-[44px] transition-[max-height] duration-500 ease-in-out peer-checked:max-h-[500px] select-none px-5 border-b-
      [1px] border-dark-bg dark:border-light-yellow bg-light-ciel/95 dark:bg-dark-bg/95 ">
        <ThemeSwitcher />
        <nav>
          <div className="flex flex-col items-end">
            <label
              id="menu-label"
              htmlFor="menu-toggle"
              className="cursor-pointer sm:hidden p-2 active:scale-95 "
            >
              <NavMenuButton />
            </label>
            <ul className="mr-[-1.25rem] sm:mr-0 gap-4 font-bold text-dark-bg dark:text-dark-grey sm:flex sm:flex-row">
            </ul>
          </div>
        </nav>
      </header>
    </>
  );
}
```

Fonctionnalité Front - Menu de navigation avec 0 JavaScript

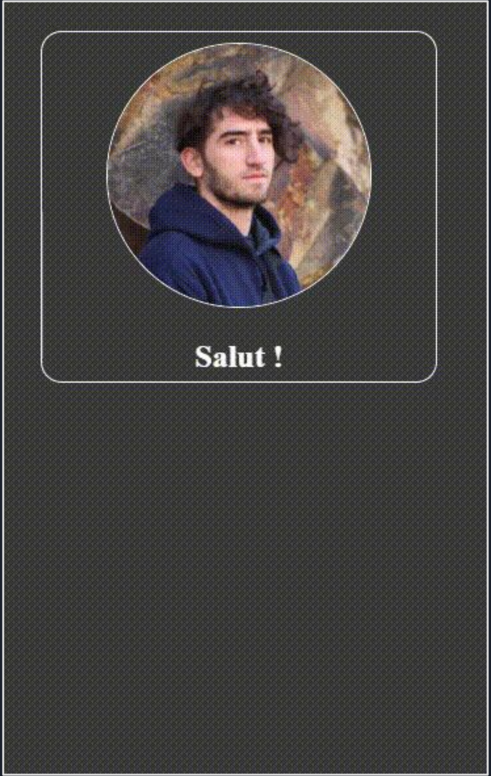
```jsx
export default async function Header() {

  return (
    <>
      <input type="checkbox" id="menu-toggle" className="hidden peer" />
      <header className="sticky z-50 top-0 min-w-full sm:max-h-none sm:items-end flex justify-between overflow-hidden max-h-[44px] transition-[max-height] duration-500 ease-in-out peer-checked:max-h-[500px] select-none px-5 border-b-[1px] border-dark-bg dark:border-light-yellow bg-light-ciel/95 dark:bg-dark-bg/95 ">
        <ThemeSwitcher />
        <nav>
          <div className="flex flex-col items-end">
            <label
              id="menu-label"
              htmlFor="menu-toggle"
              className="cursor-pointer sm:hidden p-2 active:scale-95 "
            >
              <NavMenuButton />
            </label>
            <ul className="mr-[-1.25rem] sm:mr-0 gap-4 font-bold text-dark-bg dark:text-dark-grey sm:flex sm:flex-row">
            </ul>
          </div>
        </nav>
      </header>
    </>
  );
}

```

Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```jsx
export default async function Header() {

  return (
    <>
      <input type="checkbox" id="menu-toggle" className="hidden peer" />
      <header className="sticky z-50 top-0 min-w-full sm:max-h-none sm:items-end flex justify-between overflow-hidden
      max-h-[44px] transition-[max-height] duration-500 ease-in-out peer-checked:max-h-[500px] select-none px-5 border-b-
      [1px] border-dark-bg dark:border-light-yellow bg-light-ciel/95 dark:bg-dark-bg/95 ">
        <ThemeSwitcher />
        <nav>
          <div className="flex flex-col items-end">
            <label
              id="menu-label"
              htmlFor="menu-toggle"
              className="cursor-pointer sm:hidden p-2 active:scale-95 "
            >
              <NavMenuButton />
            </label>
            <ul className="mr-[-1.25rem] sm:mr-0 gap-4 font-bold text-dark-bg dark:text-dark-grey sm:flex sm:flex-row">
            </ul>
          </div>
        </nav>
      </header>
    </>
  );
}
```

Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```css
html {
    background-color: rgb(59, 59, 59);
    color: white;
    padding: 2rem;
}
article img {
    width: 200px;
    height: 200px;
    border: solid 1px white;
    border-radius: 100%;
}
article {
    text-align: center;
    height: 250px;
    width: 300px;
    border: solid 1px white;
    border-radius: 1rem;
    padding: .5rem 0;
    overflow: hidden;
    transition: height .5s;
}
article:hover {
    height: auto;
}
```

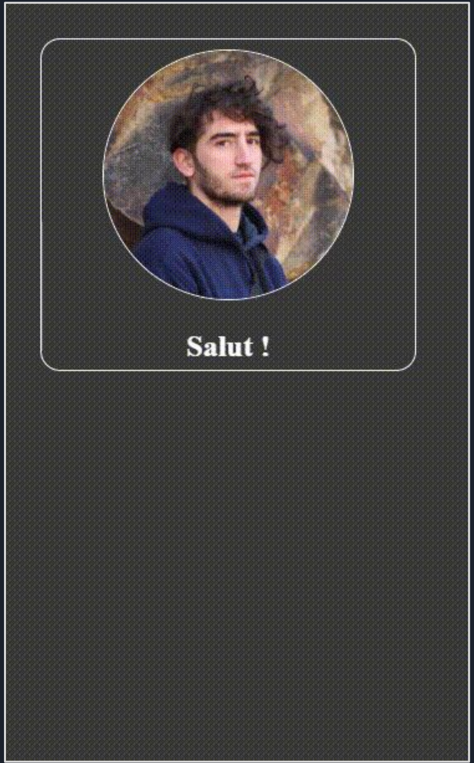


Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```css
html {
  background-color: rgb(59, 59, 59);
  color: white;
  padding: 2rem;
}
article img {
  width: 200px;
  height: 200px;
  border: solid 1px white;
  border-radius: 100%;
}
article {
  text-align: center;
  height: 250px;
  width: 300px;
  border: solid 1px white;
  border-radius: 1rem;
  padding: .5rem 0;
  overflow: hidden;
  transition: height .5s;
}
article:hover {
  height: 500px;
}
```

Fonctionnalité Front - Menu de navigation avec 0 JavaScript

```css
html {
  background-color: rgb(59, 59, 59);
  color: white;
  padding: 2rem;
}
article img {
  width: 200px;
  height: 200px;
  border: solid 1px white;
  border-radius: 100%;
}
article {
  text-align: center;
  height: 250px;
  width: 300px;
  border: solid 1px white;
  border-radius: 1rem;
  padding: .5rem 0;
  overflow: hidden;
  transition: height .5s;
  interpolate-size: allow-keywords;
}
article:hover {
  height: auto;
}
```

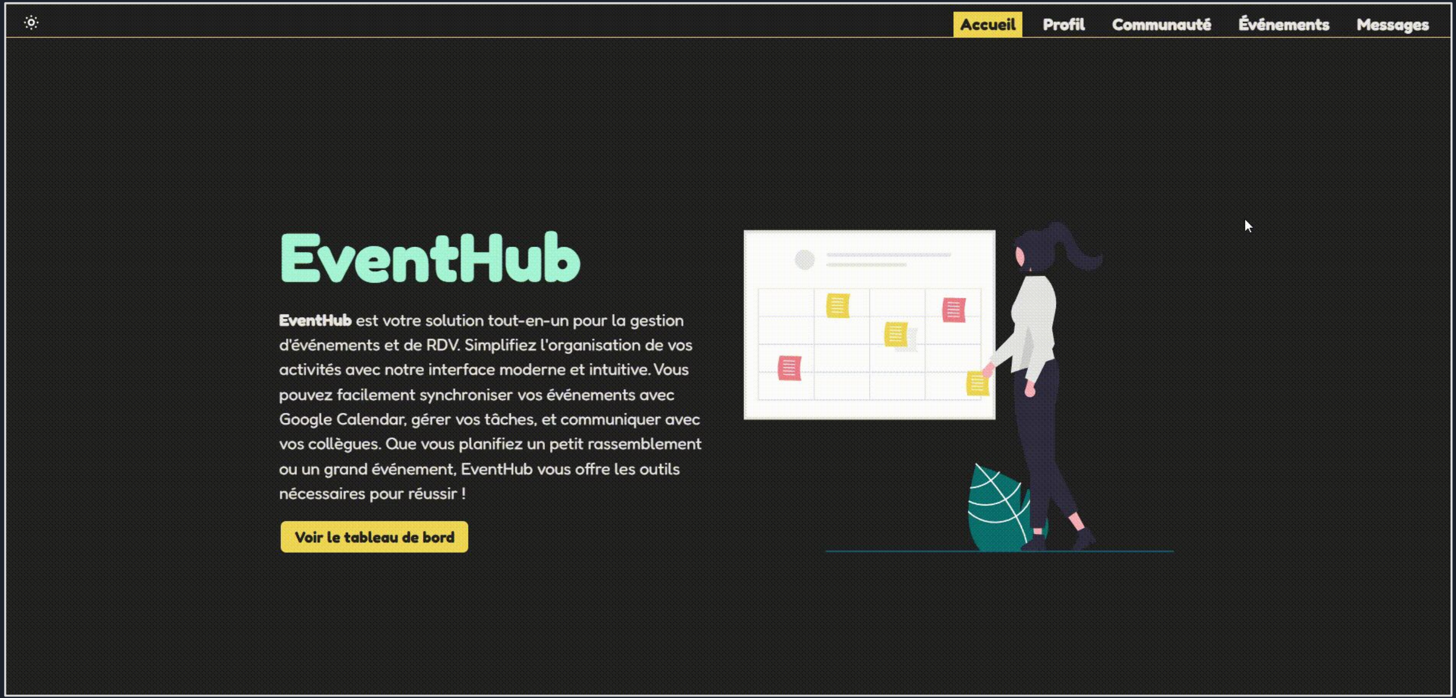


Fonctionnalité Front - Menu de navigation avec 0 JavaScript

Interpolate-size et sa compatibilité...

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
`interpolate-size`	129	No	No	115	No	129	No	No	No	No	129	No
`allow-keywords`	129	No	No	115	No	129	No	No	No	No	129	No
`numeric-only`	129	No	No	115	No	129	No	No	No	No	129	No

Fonctionnalité Front - Animation scroll



Fonctionnalité Front - Animation scroll

```
export default async function Home() {
  return (
    <main className="flex min-h-screen flex-
      <section className="animate-scroll fl
      </section>
      <section className="animate-scroll fl
      </section>
      <section className="animate-scroll fl
      </section>
    </main>
  );
}
```

```
/* ========== Scroll animation ========== */

/* Can't use this animation with Tailwind because animation-timeline is not supported by Tailwind */
@keyframes appear {
  from {
    opacity: 0;
    transform: scale(0.8);
  }
  to {
    opacity: 1;
  }
}
/* Animation applied only for large screens because the range is too big for small screens */
@media screen and (min-width: 768px) {
  .animate-scroll > h1,
  .animate-scroll > div,
  .animate-scroll > a {
    animation: appear linear;
    animation-timeline: view();
    animation-range: entry 0% cover 11%;
    animation-fill-mode: both;
    animation-duration: 1ms; /* Firefox requires this to apply the animation */
  }
}
```

Fonctionnalité Front - Animation scroll

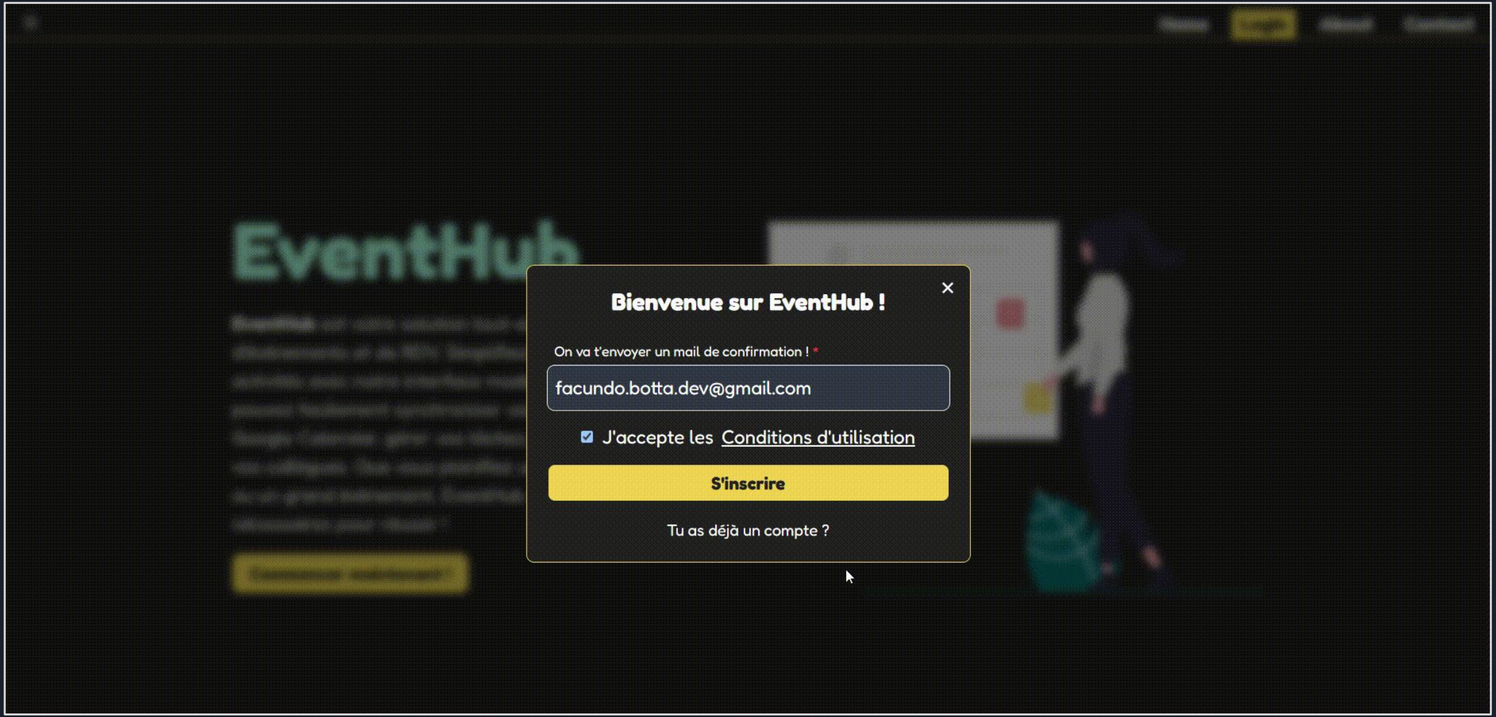
view() et sa compatibilité...

Browser compatibility

[Report problems with this compatibility data on GitHub]

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
`view()`	115	115	114	101	No *	115	No	77	No *	23.0	115	No *

Fonctionnalité Back - Création de compte avec e-mail



Fonctionnalité Back - Création de compte avec e-mail

components > forms > LogForm.tsx > LogForm

```tsx
export default function LogForm() {
  return (
    <form
      className="relative flex flex-col items-center gap-3 w-full max-w-md p-5 mx
      border-light-yellow dark:bg-dark-bg"
      onSubmit={formType === 'Sign-Up' ? handleSignUpSubmit : handleLogInSubmit}
    >
      <Icon
        type="plus" ···
      />
      <h2 className="text-2xl mb-3 font-bold text-center"> ···
      </h2>
      <Input
        className={inputClasses}
        required={true}
        label={
          formType === 'Sign-In'
            ? 'Email'
            : 'On va te envoyer un mail de confirmation !'
        }
        type="email"
        id="Email"
        name="email"
        placeholder="Email"
        disabled={isPending}
        autoComplete="email"
        errorStyles={{ color: 'red', fontSize: '1rem' }}
        error={{ message: error.mail?.message, value: error.mail?.value }}
      />
```

```tsx
const handleSignUpSubmit = (e: React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  setError({
    mail: { message: '', value: false },
    password: { message: '', value: false },
    conditions: { message: '', value: false },
  });
  if (!conditionsAccepted) {
    setError({
      ...error,
      conditions: {
        message: "Veuillez accepter les conditions d'utilisation",
        value: true,
      },
    });
    return;
  }

  try {···
  } catch (error) {···
  }
};
```

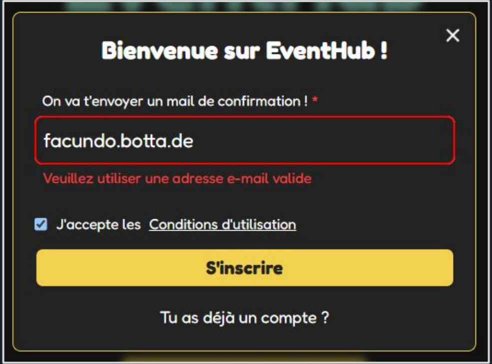
Fonctionnalité Back - Création de compte avec e-mail

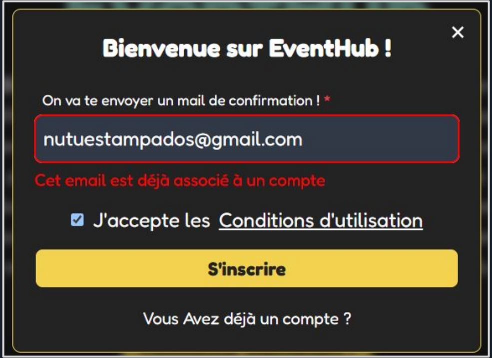
```
const handleSignUpSubmit = (e: React.FormEvent<HTMLFormElement>) => {
  e.preventDefault();
  setError({…
  });
  if (!conditionsAccepted) {…
  }

  try {
    const formData = new FormData(e.currentTarget);
    const email = getFormDataStringValue(formData, 'email');
    // validate email
    emailSchema.parse({ email });

    startTransition(async () => {
      const result = await emailSignInServerAction(email);
      if (!result?.ok) {
        setError({
          ...error,
          mail: { message: result?.message as string, value: true },
        });
      } else if (result.ok) {
        setEmailSent(true);
      }
    });
  } catch (error) {
    const handledError = handleError(error);
    setError((prevError) => ({
      ...prevError,
      mail: { message: handledError.message, value: true },
    }));
  }
};
```

```
export const emailSchema = z.object({
  email: z
    .string()
    .email({ message: 'Veuillez utiliser une adresse e-mail valide' }),
});
```

```
export default function LogForm() {
  const Router = useRouter();
  const [formType, setFormType] = useState<string>('Sign-In');
  const [isPending, startTransition] = useTransition();
  const [conditionsAccepted, setConditionsAccepted] = useState(false);
  const [error, setError] = useState({
    mail: { message: '', value: false },
    password: { message: '', value: false },
    conditions: { message: '', value: false },
  });
```

Fonctionnalité Back - Création de compte avec e-mail





Log handleError :

```
▼ handledError: CustomError: Veuillez utiliser une adresse e-mail valide at handleError (webpack-interna
  ▼ errors: Array(1)
    ▼ 0:
        message: "Veuillez utiliser une adresse e-mail valide"
      ▶ path: ['email']
      ▶ [[Prototype]]: Object
      length: 1
    ▶ [[Prototype]]: Array(0)
    name: "CustomError"
    ok: false
    message: "Veuillez utiliser une adresse e-mail valide"
    stack: "CustomError: Veuillez utiliser une adresse e-mail valide\n    at handleError (webpack-intern
  ▶ [[Prototype]]: Error
▶ [[Prototype]]: Object
```

Log result.ok == false

```
                                                          LogForm.tsx:124
▶ {ok: false, message: 'Cet email est déjà associé à un compte'}
```

Fonctionnalité Back - Création de compte avec e-mail

```
'use server';

import { signIn } from '@/lib/auth/authConfig';
import { handleError } from '@/lib/zod/handleError';
import { emailSchema } from '@/lib/zod/zodSchemas';
import { selectUserByEmailService } from '@/services/userServices';

/**
 * Handles the email sign-in process.
 *
 * This function sends an email to the provided email address with a link to sign in.
 *
 * @param {string} email - The email address to sign in with.
 * @returns {Promise<{ ok: boolean, message?: string }>} An object indicating the result of the
 sign-in process.
 */
export async function emailSignInServerAction(
  email: string
): Promise<{ ok: boolean; message: string }> {
  try {
    // Validate the email using Zod schema
    emailSchema.parse({ email });

    // Check if the user already exists
    const user = await selectUserByEmailService({ email });
    if (user) {
      return {
        ok: false,
        message: 'Un compte existe déjà avec cet email',
      };
    }

    // Proceed with the sign-in process
    await signIn('nodemailer', {
      email,
      callbackUrl: '/profile',
      redirect: false,
    });

    return { ok: true, message: 'Email envoyé' };
  } catch (error) {
    return handleError(error);
  }
}
```

```
export const selectUserByEmailService = async ({
  email,
}: {
  email: string;
}): Promise<Partial<User> | null> ⇒ {
  try {
    return prisma.user.findUnique({
      where: {
        email,
      },
      select: {
        id: true,
        role: true,
        password: true,
        hasPassword: true,
        email: true,
      },
    });
  } catch (error) {
    console.error('selectUserByEmailService', error);
    return null;
  }
};
```

Fonctionnalité Back - Création de compte avec e-mail

lib > auth > TS authConfig.ts > ...

```ts
export const { handlers, auth, signIn, signOut } = NextAuth({
  trustHost: true, // This is required for NextAuth to work properly in LocalHost
  adapter: PrismaAdapter(prisma),
  secret: process.env.AUTH_SECRET,
  session: {···
  },
  pages: {···
  },
  callbacks: {···
  },
  providers: [
    Google({···
    }),
    Nodemailer({
      server: {
        host: process.env.EMAIL_SERVER_HOST,
        port: parseInt(process.env.EMAIL_SERVER_PORT!, 10),
        auth: {
          user: process.env.EMAIL_SERVER_USER,
          pass: process.env.EMAIL_SERVER_PASSWORD,
        },
      },
      from: process.env.EMAIL_FROM,
      sendVerificationRequest: async (params) => {
        const { identifier, url, provider } = params;
        const transport = createTransport(provider.server);
        const result = await transport.sendMail({
          to: identifier,
          from: provider.from,
          subject: `Bienvenue sur EventHub !`,
          text: text({ url }), // This is the text version of the email
          html: html({ url }), // This is the HTML version of the email
        });
        const failed = result.rejected.concat(result.pending).filter(Boolean);
        if (failed.length) {
          throw new Error(`Email(s) (${failed.join(', ')}) could not be sent`);
        }
      },
    }),
    Credentials({···
    }),
  ],
});
```

.env

```
##=================================#
## MYSQL Database configuration #
##=================================#

DATABASE_URL="mysql://admin:password@localhost:3306/filrouge2024?schema=public"
DB_NAME="filrouge2024"
DB_USER="admin"
DB_PASSWORD="password"
SERVER_PORT=3306
NEXT_AUTH_URL="http://localhost:3000"

##================================#
## IDENTITY ACCESS AND MANAGEMENT #
##================================#

AUTH_GOOGLE_ID="534357650281-nklhinq8om34ht4v7hfd1gpd4hu0gm6a.apps.googleusercontent.com"
AUTH_GOOGLE_SECRET="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"
NEXTAUTH_URL="http://localhost:3000"
# generated with openssl rand -base64 33
AUTH_SECRET="g8JdX+OWEwqRHCjji2YNvJPiHxT6c2D6fdWDe6x1yIX6"

##================================#
## EMAIL SERVER CONFIGURATION     #
##================================#
EMAIL_FROM="noreply@eventhub.com"
EMAIL_SERVER_HOST="smtp.gmail.com"
EMAIL_SERVER_PORT=587
EMAIL_SERVER_USER="nutuestampados@gmail.com"
# Temporary password for adrar
EMAIL_SERVER_PASSWORD="mkmv kpbq lsnm dahp"
```

Fonctionnalité Back - Création de compte avec e-mail

lib > auth > TS authConfig.ts > ...

```ts
export const { handlers, auth, signIn, signOut } = NextAuth({
  trustHost: true, // This is required for NextAuth to work properly in localHost
  adapter: PrismaAdapter(prisma),
  secret: process.env.AUTH_SECRET,
  session: {···
  },
  pages: {···
  },
  callbacks: {···
  },
  providers: [
    Google({···
    }),
    Nodemailer({
      server: {
        host: process.env.EMAIL_SERVER_HOST,
        port: parseInt(process.env.EMAIL_SERVER_PORT!, 10),
        auth: {
          user: process.env.EMAIL_SERVER_USER,
          pass: process.env.EMAIL_SERVER_PASSWORD,
        },
      },
      from: process.env.EMAIL_FROM,
      sendVerificationRequest: async (params) => {
        const { identifier, url, provider } = params;
        const transport = createTransport(provider.server);
        const result = await transport.sendMail({
          to: identifier,
          from: provider.from,
          subject: `Bienvenue sur EventHub !`,
          text: text({ url }), // This is the text version of the email
          html: html({ url }), // This is the HTML version of the email
        });
        const failed = result.rejected.concat(result.pending).filter(Boolean);
        if (failed.length) {
          throw new Error(`Email(s) (${failed.join(', ')}) could not be sent`);
        }
      },
    }),
    Credentials({···
    }),
  ],
});
```

lib > TS emailTemplate.ts > ...

```ts
export function html(params: { url: string }) {
  const { url } = params;

  const color = {
    background: '#f9f9f9',
    text: '#444',
    mainBackground: '#fff',
    buttonBackground: '#F3D250',
    buttonBorder: 'black',
    buttonText: 'black',
  };

  return `···
`;
}

// Email Text body (fallback for email clients that don't render HTML, e.g. feature phones)
export function text({ url }: { url: string }) {
  return `Bienvenue sur EventHub !\nNous sommes ravis de vous accueillir. Pour continuer,
  cliquez sur le lien suivant pour vous connecter:\n${url}\n\nSi vous n'avez pas demandé
  cette connexion, vous pouvez ignorer cet email.\n`;
}
```

Bienvenue sur **EventHub** !

Nous sommes ravis de vous accueillir. Pour continuer, cliquez sur le bouton ci-dessous pour commencer à utiliser votre compte !.

Se connecter

http://localhost:3000/api/auth/callback/nodemailer?callbackUrl=http%3A%2F%2Flocalhost%3A3000%2Flogin&token=bdf582b70c049b817a28dabc40ea3a0cc5bcad3a46e9844cdf80d2ac489e93c5&email=facundo.botta.dev%40gmail.com

Fonctionnalité Back - Modification de compte - mdp

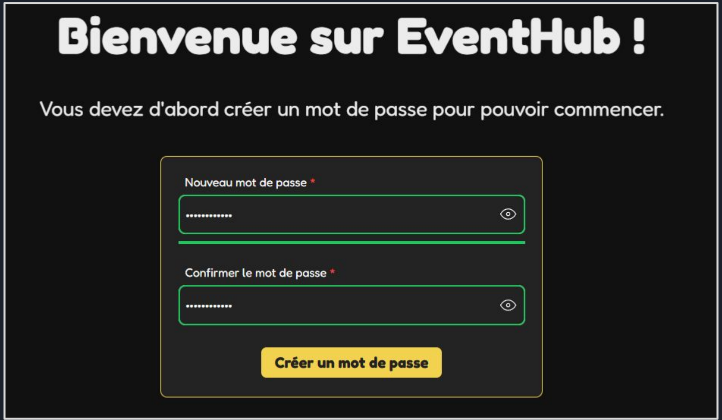
```typescript
import { getToken } from 'next-auth/jwt';
import { NextRequest, NextResponse } from 'next/server';

export async function middleware(req: NextRequest) {
  const token = await getToken({ req, secret: process.env.AUTH_SECRET });
  const isAuth = !!token;
  const hasPassword = token?.hasPassword === true;
  const isAdmin = token?.role === 'admin';
  const isAdminPage = req.nextUrl.pathname.startsWith('/admin');
  if (isAdminPage) {
    if (isAuth && isAdmin) {
      return NextResponse.next();
    } else {
      return NextResponse.redirect(new URL('/unauthorized', req.url));
    }
  }
  if (req.nextUrl.pathname.startsWith('/login') && isAuth) {
    return NextResponse.redirect(new URL('/profile', req.url));
  }
  if (
    req.nextUrl.pathname.startsWith('/profile') ||
    req.nextUrl.pathname.startsWith('/messages') ||
    req.nextUrl.pathname.startsWith('/events') ||
    req.nextUrl.pathname.startsWith('/communaute')
  ) {
    if (!isAuth) {
      return NextResponse.redirect(new URL('/login', req.url));
    }
    if (!hasPassword) {
      return NextResponse.redirect(new URL('/set-password', req.url));
    }
  }

  return NextResponse.next();
}

export const config = {
  matcher: [
    '/login',
    '/admin/:path*',
    '/profile/:path*',
    '/messages/:path*',
    '/events/:path*',
    '/communaute/:path*',
  ],
};
```

Log jeton de session

```
{
  token: {
  name: null,
  email: 'facundo.botta.dev@gmail.com',
  picture: null,
  sub: 'cm7lyag1h0002rgq8m8fimjpi',
  id: 'cm7lyag1h0002rgq8m8fimjpi',
  role: 'user',
  hasPassword: false,
  iat: 1740576482,
  exp: 1741181282,
  jti: '780a1e7f-3a77-4f06-85f5-7982c2d1c534'
}
```

Fonctionnalité Back - Modification de compte - mdp

http://localhost:3000/set-password




```
const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,30}$/;

export const passwordSchema = z.object({
  password: z
  .string()
    .min(8, { message: 'Le mot de passe doit être de 8 caractères minimum' })
    .max(30, { message: 'Le mot de passe doit être de 30 caractères maximum' })
    .regex(passwordRegex, {
      message: 'Le mot de passe doit contenir au moins 8 caractères, une majuscule, une
      minuscule, un chiffre et un caractère spécial',
    }),
});
```

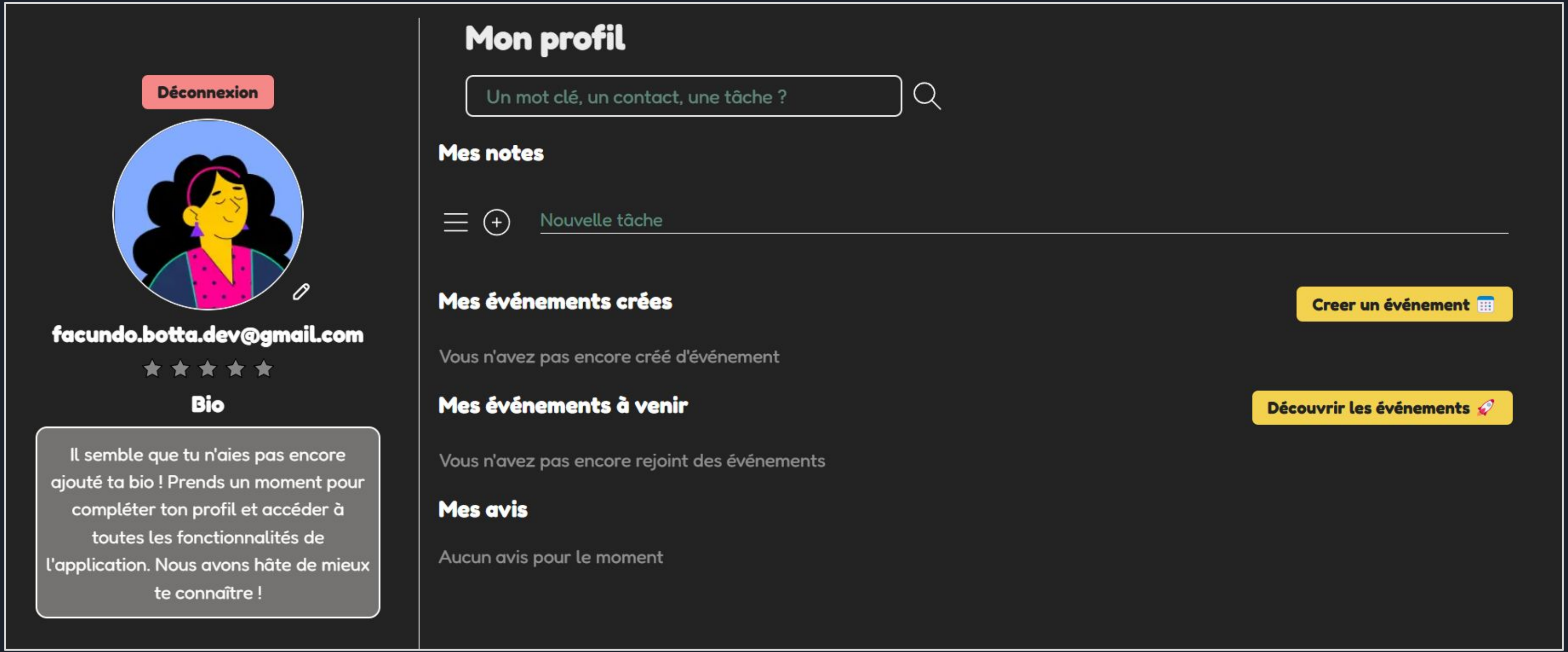
```
/**
 * Updates or create the user's password.
 *
 * @param {FormData} formData - The FormData object containing the user ID and new password.
 * @returns {Promise<{ ok: boolean, message?: string }>}
 */
export const updatePassword = async (formData: FormData) => {
  const session = await auth();

  const id = getFormDataStringValue(formData, 'id');
  const password = getFormDataStringValue(formData, 'password');

  if (!session || session.user.id !== id) {
    return {
      ok: false,
      message: 'Non connecté',
    };
  }

  async function hashPassword(password: string): Promise<string> {
    const saltRounds = 10;
    const hashedPassword = await bcrypt.hash(password, saltRounds);
    return hashedPassword;
  }

  try {
    // Validate the password using Zod schema
    passwordSchema.parse({ password });

    // Hash the password
    const hashedPassword = await hashPassword(password);

    // Update the user's password
    await updateUserService({
      id,
      data: {
        password: hashedPassword,
        hasPassword: true,
      },
    });

    return { ok: true, message: 'Mot de passe mis à jour avec succès' };
  } catch (error) {
    return handleError(error);
  }
};
```

Fonctionnalité Back - Modification de compte - mdp

```typescript
export const updateUserService = async ({
  id,
  data,
}: {
  id: string;
  data: Pick<
    Prisma.UserUpdateInput,
    'image' | 'description' | 'username' | 'hasPassword' | 'password'
  >;
}) => {
  try {
    return prisma.user.update({
      where: {
        id,
      },
      data,
    });
  } catch (error) {
    console.error('updateUserService', error);
    return null;
  }
};
```

```typescript
export type UserUpdateInput = {
  id?: StringFieldUpdateOperationsInput | string
  name?: NullableStringFieldUpdateOperationsInput | string | null
  username?: NullableStringFieldUpdateOperationsInput | string | null
  password?: NullableStringFieldUpdateOperationsInput | string | null
  hasPassword?: BoolFieldUpdateOperationsInput | boolean
  email?: NullableStringFieldUpdateOperationsInput | string | null
  emailVerified?: NullableDateTimeFieldUpdateOperationsInput | Date | string | null
  image?: NullableStringFieldUpdateOperationsInput | string | null
  description?: NullableStringFieldUpdateOperationsInput | string | null
  role?: NullableStringFieldUpdateOperationsInput | string | null
  createdAt?: DateTimeFieldUpdateOperationsInput | Date | string
  updatedAt?: DateTimeFieldUpdateOperationsInput | Date | string
  Session?: SessionUpdateManyWithoutUserNestedInput
  Account?: AccountUpdateOneWithoutUserNestedInput
  Authenticator?: AuthenticatorUpdateManyWithoutUserNestedInput
  CommentsAuthored?: CommentsUpdateManyWithoutAuthorNestedInput
  CommentsReceived?: CommentsUpdateManyWithoutRecipientNestedInput
  Reports?: ReportsUpdateManyWithoutUserNestedInput
  Reported?: ReportsUpdateManyWithoutReportedNestedInput
  Trash?: TrashUpdateManyWithoutUserNestedInput
  Ratings?: RatingsUpdateManyWithoutUserNestedInput
  EventsCreated?: EventsUpdateManyWithoutUserNestedInput
  EventsJoined?: UserEventsUpdateManyWithoutUserNestedInput
  Tasks?: TasksUpdateManyWithoutUserNestedInput
  MessagesSent?: MessageUpdateManyWithoutSenderNestedInput
  UserConversation?: UserConversationUpdateManyWithoutUserNestedInput
  MessageStatus?: MessageStatusUpdateManyWithoutUserNestedInput
  UserInvitations?: UserInvitationsUpdateManyWithoutUserNestedInput
}
```

Fonctionnalité Back - Modification de compte - mdp



Tests - updateUserService

services > TS userServices.test.ts > ...

```ts
import { PrismaClient } from '@prisma/client';
import { mockDeep } from 'jest-mock-extended';
import { expect } from '@jest/globals';
import {
  selectAllUsersService,
  selectUserByEmailService,
  selectUserByIdService,
  updateUserService,
} from './userServices';
import {
  getAllUsersServiceSchema,
  getUserServiceSchema,
} from '@/lib/zod/zodSchemas';

jest.mock('@/lib/prisma', () => ({
  __esModule: true,
  default: mockDeep<PrismaClient>(),
}));

const prismaMock = jest.requireMock('@/lib/prisma').default;

const mockUser = {
  id: '1',
  email: 'user@example.com',
  username: 'testuser',
  password: 'secretpassword',
  image: null,
  description: null,
  _count: { Ratings: 2, EventsCreated: 1 },
};
```

```ts
describe('User Services', () => {
  afterEach(() => {
    jest.clearAllMocks();
  });
  describe('UpdateUserService', () => {
    it('should update a user successfully', async () => {
      const newData = {
        username: 'newUserName',
        description: 'new description',
      };

      prismaMock.user.update.mockResolvedValue({
        ...mockUser,
        ...newData,
      });

      const result = await updateUserService({
        id: '1',
        data: newData,
      });

      expect(prismaMock.user.update).toHaveBeenCalledWith({
        where: { id: '1' },
        data: newData,
      });
      expect(result).toEqual({
        ...mockUser,
        ...newData,
      });
    });
  });
  describe('selectUserByIdService', () => {…
  });
  describe('selectUserByEmailService', () => {…
  });
  describe('selectAllUsersService', () => {…
  });
});
```

Tests - Log des tests

```
$ npm run test

> filrouge@0.1.0 test
> jest --verbose

 PASS  services/userServices.test.ts
  User Services
    UpdateUserService
      √ should update a user successfully (4 ms)
    selectUserByIdService
      √ should select a user by id successfully (zod schema ok and password not returned) (3 ms)
      √ should return null when user is not found
    selectUserByEmailService
      √ should select a user by email successfully
      √ should return null when user is not found
    selectAllUsersService
      √ should select all users successfully (1 ms)
      √ should return null when there is an error

 PASS  lib/zod/handleError.test.ts
  handleError function
    √ should handle Zod validation errors correctly (2 ms)
    √ should handle generic errors correctly (1 ms)

 PASS  utils/customError.test.ts
  CustomError class
    √ should create an instance with a message and no errors (1 ms)
    √ should create an instance with a message and errors (1 ms)
    √ should set the name property to the constructor name
    √ should inherit from the built-in Error class (1 ms)

Test Suites: 3 passed, 3 total
Tests:       13 passed, 13 total
Snapshots:   0 total
Time:        1.218 s
Ran all test suites.
```

Conclusion

À faire 📝

Bugs 🪲

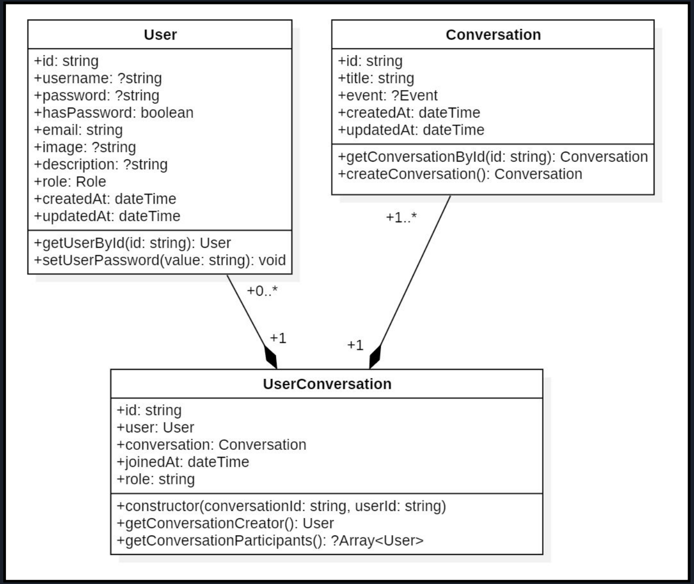
À venir 🔮

Liens :

- Figma
- Mémoire
- Portfolio
- Site développé en freelance

Merci ! 😁

Bonus - Diagramme de classe de example



Approche fonctionnelle - Services

```
export const getConversationParticipantsService = async (
  conversationId: string
) => {
  try {
    const participants = await prisma.userConversation.findMany({
      where: { conversationId },
      select: {
        userId: true,
      },
    });
    return participants;
  } catch (error) {
    console.error('getConversationService: error', error);
    throw new Error('Service error: getConversationService');
  }
};
```

const creator = new UserConversation(<<conversationId>>, <<userId>>).getConversationCreator();

Bonus - handleError

```typescript
import CustomError from '@/utils/customError';
import { ZodError } from 'zod';

// This interface is used by the handleError function to type the error object
export interface ValidationError {
  message: string;
  path: (string | number)[];
}

/**
 * Handles errors and converts them into a standardized `CustomError` instance.
 * Supports both generic errors and `ZodError` validation errors.
 * *
 * @param {unknown} error - The error to handle. This can be any type of error, including validation errors from Zod.
 * @returns {CustomError<ValidationError[]>} A structured `CustomError` instance containing:
 *   - `message`: A general error message, which may describe validation errors or a generic issue.
 *   - `errors`: An array of validation errors if the error is a `ZodError`; otherwise, it is undefined.
 *
 * @example
 * try {
 *   schema.parse(data);
 * } catch (error) {
 *   return handleError(error);
 * }
 */
export function handleError(error: unknown): CustomError<ValidationError[]> {
  if (error instanceof ZodError) {
    // Extract validation errors from Zod
    const validationErrors: ValidationError[] = error.errors.map(
      ({ message, path }) => ({
        message,
        path,
      })
    );

    // Create an instance of CustomError with the validation error details
    const message = validationErrors.map((e) => e.message).join(', ');
    return new CustomError<ValidationError[]>(message, validationErrors);
  }

  // For other errors, generate a CustomError with a generic message
  return new CustomError<ValidationError[]>('Une erreur est survenue');
}
```

```
C:\Users\User\Docume
▼ {errorHandled: CustomError: Le mot de passe doit être de 8 caractères minimum, 
  uscule, une mi…} i
  ▼ errorHandled: CustomError: Le mot de passe doit être de 8 caractères minimum,
    ▼ errors: Array(2)
      ▼ 0:
          message: "Le mot de passe doit être de 8 caractères minimum"
        ▶ path: ['password']
        ▶ [[Prototype]]: Object
      ▼ 1:
          message: "8 caractères, une majuscule, une minuscule, un chiffre et un c
        ▶ path: ['password']
        ▶ [[Prototype]]: Object
        length: 2
      ▶ [[Prototype]]: Array(0)
      name: "CustomError"
      ok: false
      message: "Le mot de passe doit être de 8 caractères minimum, 8 caractères, u
      stack: "CustomError: Le mot de passe doit être de 8 caractères minimum, 8 ca
    ▶ [[Prototype]]: Error
  ▶ [[Prototype]]: Object
```