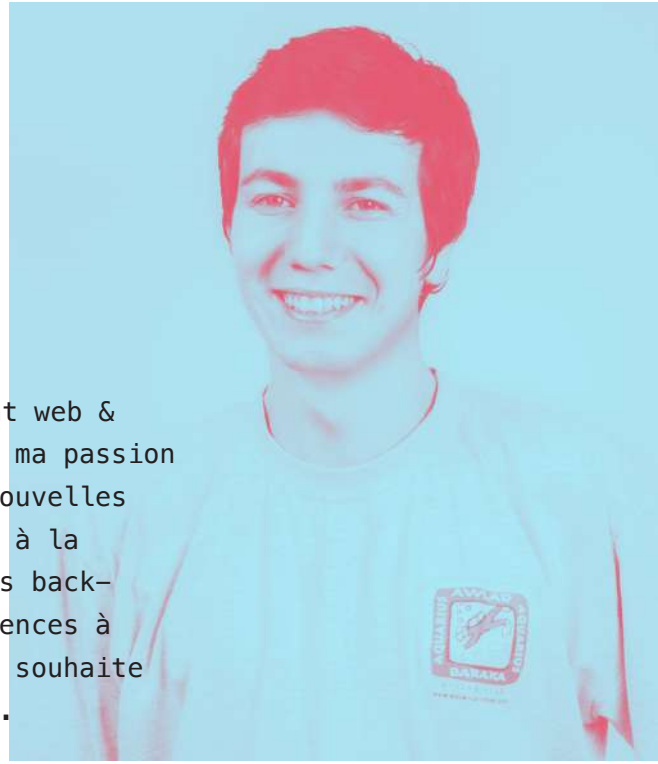
DOSSIER TECHNIQUE

Je m’appelle Bastien Robert et je suis développeur web

Sommaire

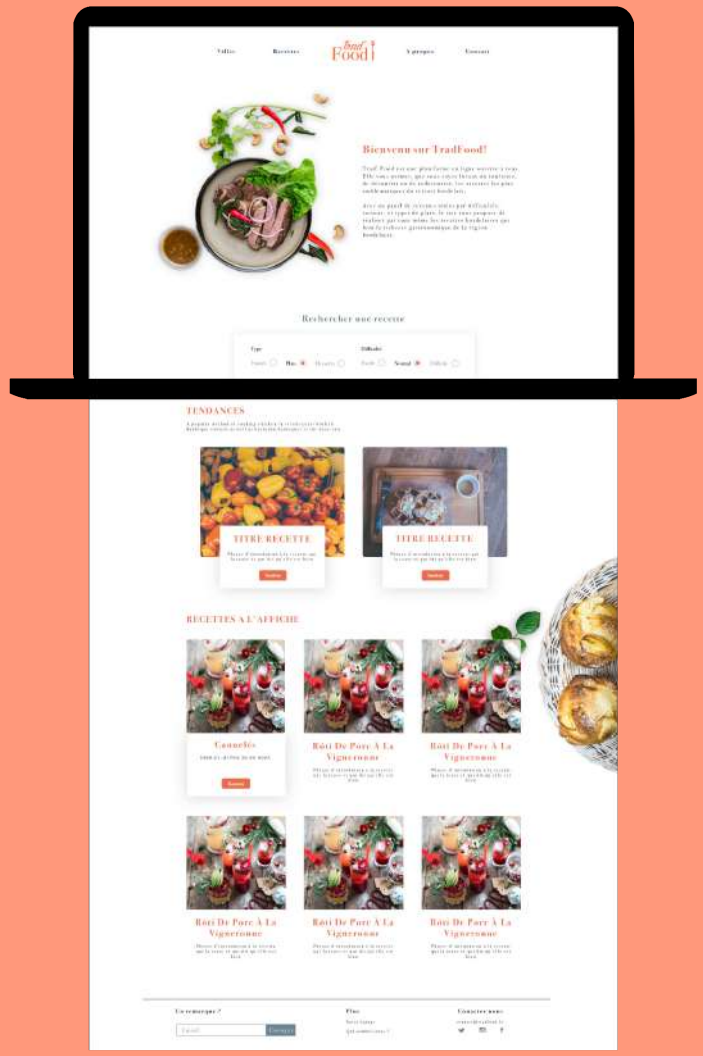
Je m’appelle Bastien et je suis étudiant en développement web & mobile en seconde année de DUT MMI à Bordeaux. Porté par ma passion pour Internet, j’essaie d’apprendre et de découvrir de nouvelles technologies chaque jour. Depuis tout jeune, je m’adonne à la création de sites web des langages front-end aux langages back-end. Lors de ma formation, j’ai pu développer mes compétences à travers plusieurs projets web et mobiles. Aujourd’hui je souhaite me consacrer au développement d’expériences interactives.



1 TRADFOOD

tradfood.fr github.com/tradfood

HTML, SASS, ES2015, VueJS, Postgres, Ruby on Rails, Dokku



Tradfood est une plateforme web regroupant les **recettes** les plus emblématiques du terroir bordelais.
Le site propose de nombreuses recettes qu'il est possible de trier par niveau de difficulté, saison et type de plats.

Pour le développement front-end de l'application, j'ai choisi d'utiliser VueJS afin d'améliorer le temps de chargement du site. L'ancienne version, basée uniquement sur une application Rails, chargeait en environ 3 secondes ; avec VueJS le chargement est passé à 1 seconde.

Fonctionnalités

API : TradFood dispose d’une API REST qui permet de récupérer : la liste des villes, une ville et ses recettes ou simplement une recette

Recherche filtrée : il est possible pour le visiteur de trier les recettes par paramètres de difficulté ou de type de plat

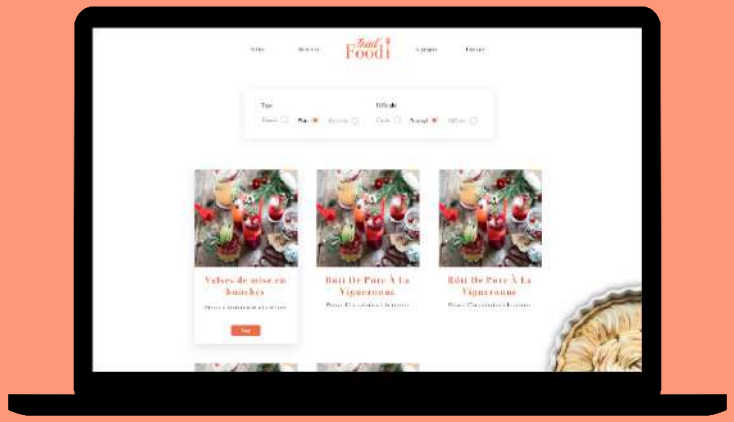
Print : bien entendu, il est possible d’imprimer une recette grâce au CSS adapté au print

1 TRADFOOD

tradfood.fr github.com/tradfood

TradFood est découpé en **2 applications :**
- la première, réalisée en **Ruby on Rails**, chargée de la partie back-end
- la seconde, réalisée avec le boilerplate **VueJS** « Bourgeon » (écrit par Ray Franco)

L'**API** construite avec Rails permet de récupérer et de traiter les informations pour les injecter dans le **DOM** de l'application VueJS. Le chargement en **asynchrone** améliore les performances et réduit la durée d'affichage de la page.



D

Recherche filtrée

TradFood permet de **trier** les recettes par type de plats, et par difficulté. J'ai donc écrit une méthode afin de trier les recettes en fonction des paramètres choisis.

```
methods: {
  sort () {
    let filteredRecipes = []
    this.city.recipes.map((k, v) => {
      this.filter(k) ? filteredRecipes.push(k) : null
    })
    this.recipes = filteredRecipes
  },
  filter (v) {
    if ((v.difficulty === parseInt(this.filtersData.difficulty) ||
      this.filtersData.difficulty === 'all') &&
      (v.meal_type === this.filtersData.mealType || this.filtersData.mealType === 'all')) {
        return true
    }
    return false
  }
}
```

La première méthode **sort** s'occupe de mapper toutes les recettes, puis les envoie à une méthode **filter.** Cette méthode vérifie que les paramètres de la recette correspondent bien à ceux de la recherche. Enfin, elle retourne un booléen qui permet d'injecter ou non la recette dans un tableau, qui sera ensuite renvoyé vers le DOM pour que les recettes sélectionnées soient affichées.

Vue resources (API)

TradFood exploite une **API** construite avec Ruby on Rails, puis rapatrie les informations des villes et des recettes sur l'application front-end **VueJS** pour les traiter et les mettre en forme.

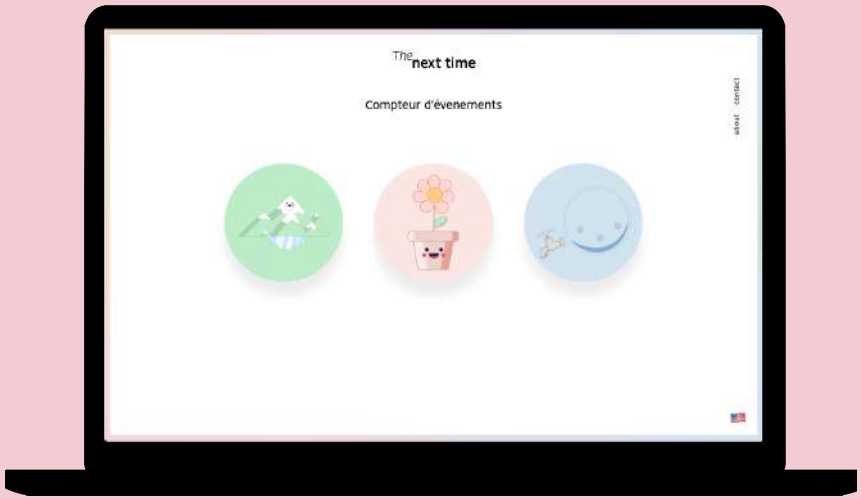
```
export default {
  data () {
    return {
      recipe: []
    }
  },
  mounted () {
    const url = 'https://api.tradfood.fr/' + this.$route.params.city +
                '/' + this.$route.params.recipe + '.json'
    this.$http.get(url).then(response => {
      this.recipe = response.body
    }, response => {
      console.log(response)
    })
  }
}
```

2 NEXT TI

next-time.cf github.com/next-time/core

HTML, SASS, ES2015, VueJS, AnimeJS, Github pages

Next Time est un site web proposant des **comptes à rebours** pour les différentes saisons de l'année. Uniquement réalisé en JavaScript, chaque compteur est accompagné d'une d'illustration vectorielle, animée avec la librairie AnimeJS.



Fonctionnalités

Animations : une fois les illustrations vectorielles réalisées par l'équipe de design, j'ai utilisé la librairie AnimeJS pour animer les différents éléments

Counter : chaque dessin s'accompagne d'un compte à rebours, qui permet d'informer l'utilisateur de l'intervalle de temps entre le jour où il consulte le site et la prochaine date de l'événement

Traduction : le site est traduit en anglais et en français grâce à vue-i18n

ME

Counter en JS

Pour chaque saison, un **compteur** est directement intégré au **composant.** Afin d'éviter la **redondance** de code et pour faciliter la **maintenance,** j'ai créé un composant tiers qui est appelé dans les différentes vues des saisons.

```vue
<template>
  <div id="counterdown" :style="{color: color}">
    <p>
      // Display days, hours and minutes
      <b>{{ seconds | two_digits }}</b>
      <em>{{ $t('counter.seconds') }}</em>
    </p>
    <div id="until">
      <em v-if="this.quote && lang == 'fr'">{{ $t('counter.untilQuote') }}</em>
      <em v-else>{{ $t('counter.until') }}</em>
      <b>{{ $t(this.until) }}</b>
    </div>
  </div>
</template>
```

Différents **paramètres** peuvent être envoyés lors de l'appel du composant :

```vue
<Countdown date="June 21, 2018" until="seasons.summer" quote color="#fceed5"></Countdown>
```

- Tout d'abord la **date,** puis la **clé** (until) i18n renvoyant vers le texte à écrire
- Le paramètre **quote** modifie le texte français "Avant...", il ajoute une apostrophe en fonction du texte (par exemple : *Avant l'été* aura le paramètre quote valorisé à true alors que *Avant le printemps* ne l'aura pas).
- Et enfin, le paramètre **color** permet de changer la couleur du texte en fonction de la saison.

next-time.cf github.com/next-time/core

AnimeJS

Nous incluons donc le **composant** qui nous permet d'afficher le **compteur** et le **sourire**, puis le code SVG.
L'intégralité du style du composant est directement stockée dans le même fichier et écrite en **stylus.**

Lors du **mounted**, nous lançons tour à tour les **animations AnimeJS**, chacune étant décrite dans une méthode spécifique.

```
<template>
  <div>
    <Countdown date="March 20, 2018" until="seasons.spring" color="#fed2cb"></Countdown>
    <div id="animation">
      <svg version="1.1" id="main" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px
        viewBox="0 0 301.6 553.6" style="enable-background:new 0 0 301.6 553.6;" xml:space="preserve">
        <g id="leaf">
          <path class="main" d="M154.6,278c18.2-28.1,43.4-33.8,57.6-34.6c10-0.6,18.2,8.3,16.1,18.2c-0.4,1.8-1,3.7-1.9,5.7
            c-6.9,15-12.9,30.9-71.1,41.1"/>
          <path class="shadow" d="M222.9,248c0.6,2,0.9,3.8,0.5,6c-0.3,1.6-0.9,3.3-1.7,5.2c-6.2,13.5-24.9,31.4-64.3,47.7
            c56.3-10,62.1-25.3,68.7-39.8c0.9-2,1.5-3.8,1.8-5.5C229,256.1,226.8,251.2,222.9,248z"/>
        </g>
      </svg>
    </div>
    </div>
    <Face></Face>
  </div>
</template>

<script>
import Countdown from '../partials/Countdown.vue';
import Face from '../partials/Face.vue';

export default {
  components: { Countdown, Face },
  mounted: function () { this.leaf() },
  methods: {
    leaf: function () {
      anime({
        targets: '#main #leaf',
        rotateY: ['-90deg', 0],
        duration: 2200,
        delay: 3800
      })
    }
  }
}
</script>

<style lang='stylus' scoped>
#animation
  #main
    height 50%
    min-width 500px
    position absolute
    z-index -2
    transform translateX(-50%)
    left 50%
    bottom 5vh
    #leaf
      transform-origin 0.01% 50%
      transform-box fill-box
      .main
        fill #B5EAC2
        stroke #B66A5C
        stroke-width 2
        stroke-miterlimit 10
```

ME

Traduction

La traduction du site est gérée avec **Vue-i18n.**
Les fichiers contenant les différentes **traductions** sont situés dans un répertoire dédié (un fichier par langue).
Next Time étant uniquement disponible en **français** et en **anglais,** seuls les fichiers fr.yml et en.yml sont présents dans le répertoire.

```yaml
# en.yml
messages:
  welcome: Counter of events

# fr.yml
messages:
  welcome: Compteur d'évenements
```

Il est possible d'appeler le contenu du message **welcome** dans l'objet **messages** directement dans le DOM : **{{ $t('messages.welcome') }}.** Le fait que le message soit affiché en français ou en anglais dépend de la valeur de **lang,** qui peut être édité via un simple lien :

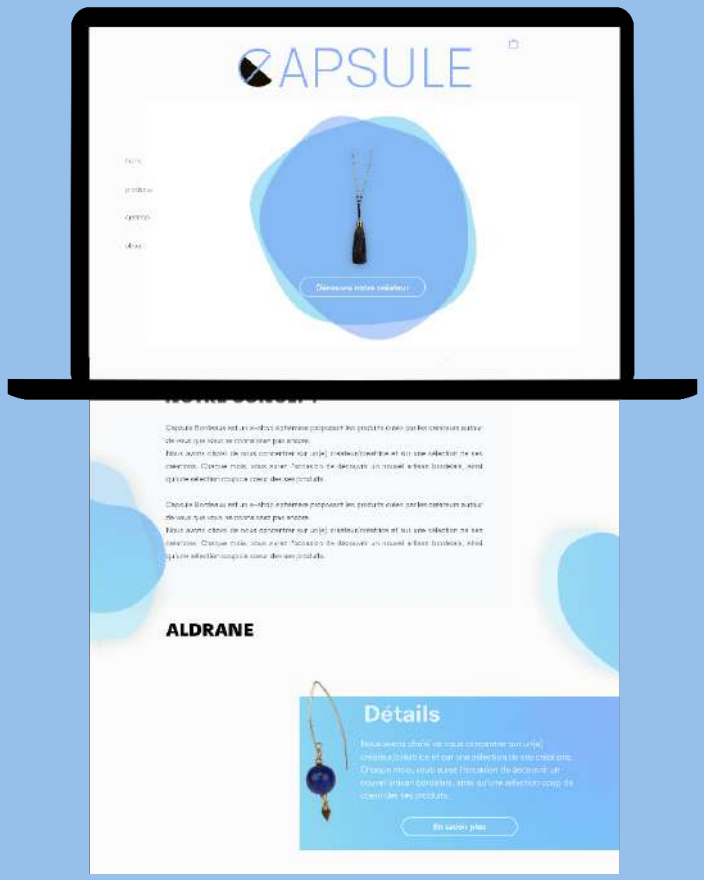
```html
<div id="lang">
  <a href="#" @click="setLang('fr')" v-show="isLang('en')">🇫🇷</a>
  <a href="#" @click="setLang('en')" v-show="isLang('fr')">🇺🇸</a>
</div>
```

Ceci permet d'afficher le drapeau français si le site est en anglais et vice-versa.

3 CAPSULE

capsule-bordeaux.fr github.com/la-capsule

HTML, SASS, ES2015, Middleman, Snipcart, Postgres, Ruby on Rails, Netlify, Heroku



Capsule est un **e-shop éphémère** proposant des produits imaginés par des créateurs de la région de Bordeaux. Chaque mois, le projet donne à un nouveau créateur la possibilité de présenter une sélection de ses créations, qui sont également disponibles à la vente directement sur la plateforme.

Capsule est découpé en 2 applications :

- la première, réalisée avec le **framework statique ruby Middleman**, chargée de la partie e-commerce
- la seconde, réalisée avec **Ruby on Rails** gère la partie administration du site, pour le créateur

La solution e-commerce utilisée par Capsule est **Snipcart**, qui est une solution Javascript all-include.

Fonctionnalités

Acheter des produits : Capsule étant une plateforme d’e-commerce, il est possible d’acheter et de se faire livrer les produits présentés

Personnaliser son produit : il est possible de choisir entre plusieurs déclinaisons de produits, la couleur, la taille ou encore le nombre d’articles que l’on souhaite ajouter au panier

Espace d’administration et API : cette fonctionnalité est en réalité une application Rails tiers, qui permet au créateur du mois de gérer ses commandes ; l’application génère une API REST qui permet, dans un deuxième temps, de rapatrier des informations pour le suivi de commande

Suivre sa commande : grâce à l’API générée dans l’application Rails d’administration pour les créateurs, il est possible de suivre sa commande directement via le site d’e-commerce statique

3 CAPSULE

capsule-bordeaux.fr github.com/la-capsule

E-commerce : Snipcart

Snipcart est une solution complète, très simple à mettre en place : pour l'installer, il suffit d'insérer le **script** et la **feuille de style ;** un simple bouton associé à la classe **snipcart-add-item** qui contient les différents **paramètres** du produit permet d'ajouter ce dernier au panier. Pour des produits customisables il faut modifier les **dataset** du bouton et remplacer les valeurs par ce que l'utilisateur choisit.

```
class Product {
  constructor () {
    this.product = document.getElementById('product')
    if (this.product != null) {
      this.productId = this.product.dataset.itemId
      this.buyButton = this.product.querySelector('#buy-' + this.productId)
      this.buyButtonData = this.buyButton.dataset
      this.pickers = this.product.querySelectorAll('form.option-picker')
      // ...
      this.buyButtonDataCustom = this.componentsConstructor()
      this.init()
    }
  }
  componentsConstructor () {
    let forms = this.product.getElementsByTagName('form')
    let customData = {}
    for (let k in this.buyButtonData) {
      if (k.includes('itemCustom') == true) {
        customData[k] = this.buyButtonData[k]
      }
    }
    return customData
  }
  init () {
    if (this.pickers != null) {
      this.quantityToggle()
    }
    for (let i = 0; i < this.pickers.length; i++) {
      this.pickers[i].addEventListener('change', () => {
        let values = this.serializeForm(this.pickers[i])
        this.detectEditItem(values)
      })
    }
  }
  serializeForm (form) {
    var els = form.elements
```

Les différents composants de personnalisation du produit sont écoutés. En cas de changement de valeur, on modifie l'élément **data-item-custom*N*-value** du bouton d'achat. Ainsi, Snipcart est notifié du changement dans la personnalisation du produit.

Suivi de commande avec l'API

Afin de **suivre la commande,** une requête est envoyée à l'**API** de **l'application d'administration ;** les informations sont ensuite rapatriées de manière asynchrone vers le DOM.
L'application d'administration **communique avec l'API de Snipcart.** Elle permet aux créateurs de consulter et d'administrer les **commandes.** Cette application génère également une API qui renvoie certains détails de la commande pour les exploiter.

```
class FollowOrder {
  constructor () {
    this.keyword = 'order'
    this.ask = document.getElementById(`follow-${this.keyword}-ask`)
    this.form = document.getElementById(`follow-${this.keyword}-form`)
    if (this.form != null) {
      this.order = this.form.querySelector("input[name='id']")
      this.response = document.getElementById(`follow-${this.keyword}-response`)
      this.responseProduct = this.response.querySelector('table.products tbody')
      this.form.addEventListener('submit', this.getOrder.bind(this))
    }
  }
  getOrder () {
    event.preventDefault()
    fetch(`https://capsule-admin.herokuapp.com/order/${this.order.value}.json`).then((response) => {
      response.json().then((json) => {
        this.pageLayout(json)
      })
    })
  }
  pageLayout (json) {
    // Insère les informations du suivi de commande dans le DOM et cache le formulaire
  }
  dataLayout (k) {
    let result = k.replace('order', '')
                  .replace(/([A-Z])/g, '-$1')
                  .substr(1)
                  .toLowerCase()
                  .split("-")

    for (let i = 0; i < result.length; i++) {
      result[i] = [result[i]]
    }

    return result
  }
  getData (json, k) {
    let data = json
    for (let i = 0; i < k.length; i++) {
      data = data[k[i]]
    }
    return data
  }
  productList (products) {
    // Obtient la liste des produits
  }
  addProduct () {
    let el = this.responseProduct.querySelector('tr').cloneNode(true)
    this.responseProduct.appendChild(el)
    return el
  }
  productLayout (p, el) {
    // Insère les informations produit dans le DOM
  }
}

export default new FollowOrder
```

Les informations de la commande sont récupérées via l'**API** de l'application d'administration en appelant la méthode **Fetch ;** elles sont ensuite envoyées à différentes fonctions qui parsent puis mettent en forme les données dans le DOM.

4 TEXPOSOU

texposou.cf github.com/bastienrobert/texposou

HTML, SASS, JS, Bootstrap, Postgres, Ruby on Rails, AWS, Heroku

Texposoù est un projet réalisé dans le cadre d'un workshop. J'ai participé au développement de cette plateforme qui vise à **mettre en relation des artistes émergents et des lieux d'exposition,** particuliers ou professionnels.

Afin de développer cette application de manière optimisée, nous nous sommes tournés vers le framework **Ruby on Rails.**

L'utilisation de **Bootstrap** pour la partie front-end nous était imposée dans le cahier des charges. Bootstrap nous a permis de développer les **wireframes** très rapidement ; toutefois, elle nous a rapidement bridés dans le développement des maquettes finales, plus complexes. Nous avons finalement utilisé nos compétences en **SCSS** afin de développer ou de modifier les composants directement dans le code.

Fonctionnalités

Espace utilisateur : un espace utilisateur complet a été mis en place pour que les utilisateurs puissent échanger. Il existe plusieurs profils d'utilisateurs : l'administrateur, le créateur, l'exposant et le visiteur

Moteur de recherche : cette fonctionnalité, à l'origine basée sur Elasticsearch, permet d'identifier un lieu ou un créateur à partir de paramètres avancés

Afficher des œuvres d'artistes et en découvrir : cette fonctionnalité affiche une page sur laquelle tous les artistes référencés sont listés et peuvent être découverts par les visiteurs du site

Découvrir des lieux d'exposition : comme pour les œuvres et les artistes, il est possible de découvrir des lieux d'exposition, disponibles ou déjà exploités par un artiste

4 TEXPOSOU

texposou.cf github.com/bastienrobert/texposou

Moteur de recherche

Texposoù permet de **rechercher** des artistes par ville ou par nom et de rechercher des lieux d'exposition par ville.

Nous nous sommes d'abord orientés vers un stack **Elastic** avec l'utilisation d'**Elasticsearch** puis nous avons finalement décidé de développer une solution plus simple à **maintenir** :

```
<div class="col-md-9">
  <% testUser = false %>

  <section class="grid-body">
    <% @users.each do |user| %>
    <% if user.avatar.exists? %>
      <% testUser = true %>
      <%= link_to show_by_id_path(user.id) do %>
      <article class="grid-item">
        <%= image_tag user.avatar.url %>
        <div class="content">
          <p class="title"><%= user.firstname %> <%= user.lastname %></p>
        </div>
      </article>
      <% end %>
      <% end %>
    <% end %>
  </section>
  <% if !testUser %>
  <p>Il n'y a pas d'artiste correspondant à votre recherche</p>
  <% end %>
</div>
```

Une variable **testUser** est générée, puis une liste d'utilisateurs correspondant à la **recherche** est affichée. Si les **utilisateurs** n'ont pas d'avatar, ils ne sont **pas affichés** car ils ne respectent pas la **charte d'inscription** qui oblige à en avoir un.

Gérer les permissions avec Cancancan

Afin de **gérer les permissions**, Texposoù utilise la gem **cancancan** qui génère un nouveau **model ability.rb** contenant les permissions pour les différents profils d'utilisateurs :

```ruby
class Ability
  include CanCan::Ability

  def initialize(user)
    # Define abilities for the passed in user here. For example:


    user ||= User.new # guest user (not logged in)
    if user.admin?
      can :access, :rails_admin       # only allow admin users to access Rails Admin
      can :dashboard
      can :manage, :all               # allow access to dashboar
    else
      #For the moment, user manage all except rails_admin
      can :manage, [
        ArtTag,
        Exhibition,
        ExhibitionsPart,
        ImagePlace,
        ImageUser,
        Participation,
        Place,
        User,
        UsersTagPart,
        Visit
      ]

    end
  end
end
```

L'utilisateur peut gérer tous les contrôleurs excepté **rails_admin** qui est l'espace d'administration uniquement accessible par le profil **administrateur.**
Dans chaque contrôleur, on spécifie les **droits** de l'utilisateur :

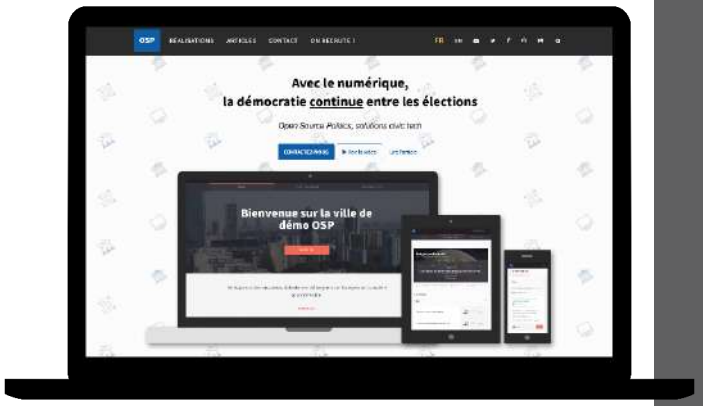
```ruby
class ProfileController < Devise::RegistrationsController
  before_action :set_user, only: [:show_profile, :edit_profile, :update_profile]
  prepend_before_action :authenticate_scope!, only: [:edit, :update, :destroy, :show_profile, :update_profile]
  prepend_before_action :set_minimum_password_length, only: [:new, :edit, :show_profile]
  before_action :authenticate_user!, except: [:index_by_status, :show_by_id]

  # Controller methods

  private
  # Use callbacks to share common setup or constraints between actions.
  def set_user
    if current_user
      @user = current_user
    else
      flash[:notice] = "Vous devez vous authentifiez avec d'acceder a cette section"
      redirect_to root_path
    end
  end

  def configure_permitted_parameters
    devise_parameter_sanitizer.permit(:update_profile) { |u| u.permit(:firstname, :lastname, :address, :address, ...)}
  end

  def sign_up_params
    params.require(:user).permit(:firstname, :lastname, :email, :password, :password_confirmation, :main_status)
  end

  def profile_params
    params.require(:user).permit(:firstname, :lastname, :address, :city, :zipcode, :firstname, :tel, :website, :bio, ...)}
  end
end
```

Si l'utilisateur est **propriétaire** du contenu, il a accès au **formulaire d'édition** et à la **méthode de suppression ;** sinon il est redirigé vers la page d'accueil et un message lui **notifie** qu'il n'a pas les autorisations nécessaires.

5 SIDE PRO

opensourcepolitics.eu *github.com/openSourcePolitics*

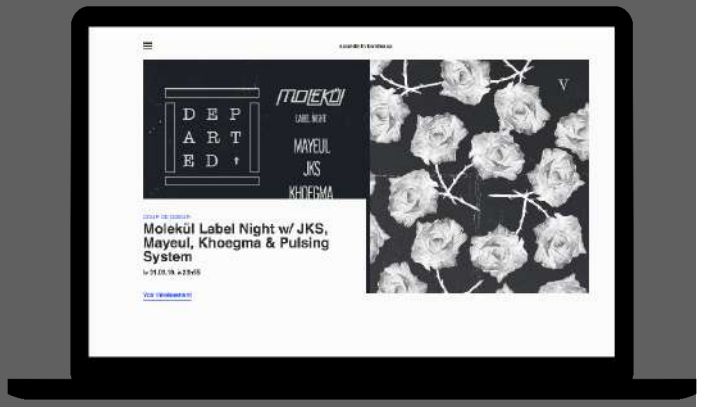
Decidim est un projet Open Source qui vise à développer la participation citoyenne dans diverses organisations (villes, départements, régions…). J'ai participé au développement de Decidim en liaison avec la société qui édite ce logiciel : Open Source Politics.



OJETS

soundsinbordeaux.com *github.com/emelinebailleul/soundsinbdx-v2*

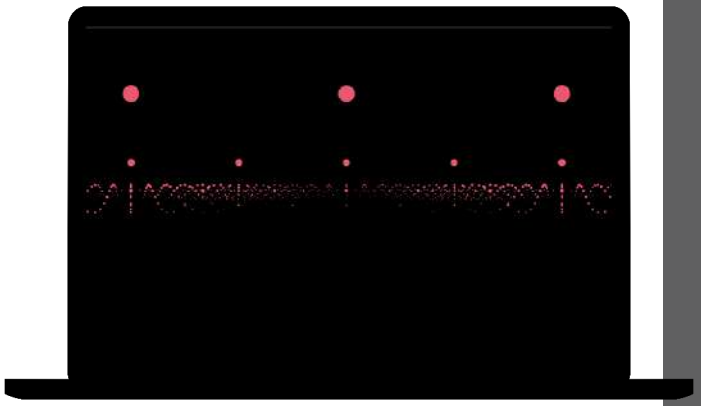
Sounds In Bordeaux est une initiative étudiante qui vise à faire connaître les artistes et les événements de musique électronique à Bordeaux. L’objectif est de favoriser la découverte, de promouvoir les soirées et de proposer un agenda des meilleures sorties à venir sur la scène bordelaise.
J’ai participé au développement de la seconde version, réalisée avec le framework Ruby on Rails.



5 SIDE PRO

codepen.io/bastienrobert/pen/aEEBEQ

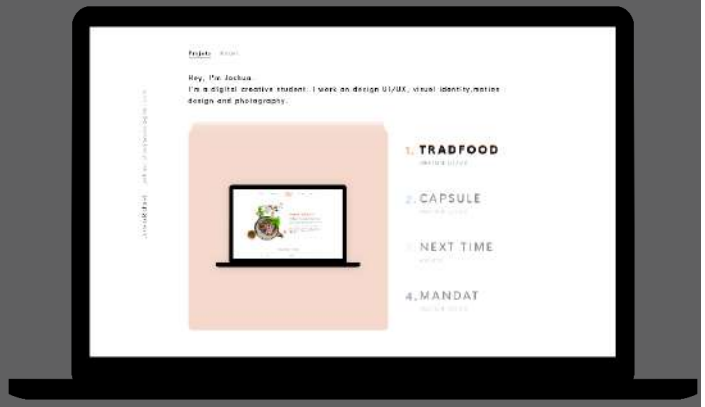
Après m’être intéressé à la Web Audio API, le WebGL natif puis à la librairie ThreeJS, je me suis inspiré de plusieurs animations basées sur du son pour réaliser la mienne.



OJETS

joshuarichard.fr *github.com/bastienrobert/joshuarichard*

En dernière année de DUT, j’ai participé à la réalisation du portfolio de Joshua Richard avec qui j’ai collaboré pour de nombreux projets durant ma formation MMI. Naturellement, nous avons travaillé ensemble afin de construire nos identités graphiques mutuelles. Afin de faciliter le développement de nos deux portfolios, j’ai développé un starter Middleman open-source intégrant les fonctionnalités et l’architecture de base. Ce starter a été forké par d’autres personnes de ma promotion.



Say hello !

bastienrobert.fr