



Symfony

---

# Projet Go Eat

“ L’Uber Eats français réalisé en Symfony. ”



CDA 2022 - Projet Symfony

Semaine du 7 février au 11 février 2022

---

Amélie Lemaire • Anaïs Torres • Charles Choquet • Dany Rose

# GESTION DE PROJET



# Organisation du projet

**Différents outils combinés à diverses pratiques**

→ **Trello**

Outils de gestion de projet sous forme de tableau de tâches à organiser

→ **Communication**

Débriefs réguliers entre nous, notamment le matin

→ **Forces et faiblesses**

Prise en compte des forces et faiblesses de chacun

# Trello du groupe

The image shows a Trello board titled "Trello du groupe" with a background image of a snowy landscape and a bare tree. The board is organized into six columns, each representing a stage in the project workflow:

- Modèles**: Contains cards for "BDD", "Dossier", "Back", "Front", and "Git", each with a label "Cette carte est un modèle." and a plus icon to add more cards.
- À faire**: Contains cards for "Gestions de la liste des restaurants", "Support technique", "Barre de recherche", "Page détails resto", "Details du plat", "Details d'un resto", "Page d'informations", and "page mes infos".
- En cours**: Contains cards for "Dossier Technique", "Gestion des commandes", "Entité statut", "Page commande", "UML", and "Inscription".
- Révision du code**: Contains cards for "Dico de données", "MCD", "Login", "Gestion de la liste des plats", "Création de la page d'accueil", "CRUD détail commande", and "CRUD Restaurateurs".
- Test**: Contains a card for "Inscription générale".
- Terminé**: Contains a card for "Dossier Organisation, planification".

Each card has a title, a description, and a status icon (e.g., "C" for "En cours", "AT" for "À faire", "AL" for "À l'étude"). The "Inscription" card in the "En cours" column has a comment icon and the number "1". The "CRUD détail commande" card in the "Révision du code" column has a plus icon to add more cards.

# Organisation

- Méthode Agile
- Débriefs réguliers
- Stand-up quotidiens

## Méthode Agile

“ On parle de la méthode Agile la première fois en 1986, mais c'est seulement en 1993 qu'on la mettra enfin en œuvre et en 2001 le manifeste Agile verra le jour. Aujourd'hui, une grande partie des entreprises de développement travaillent en s'organisant à l'aide de la méthode Agile. ”

# Planning prévisionnel (sprint d'une semaine)

## Lundi

Mise en place de l'organisation et élaboration de la base de données

## Dernière matinée

Révision du code, lecture du dossier écrit et du support de présentation

Semaine du 7 février au 11 février

## Tout au long de la semaine

Elaboration du front et du back, programmation pure


# Différentes interfaces

→ Hors connexion

→ Client

→ Restaurateur

→ Livreur



Nommage des branches GitHub en fonction des tâches. GE pour Go Eat, B pour le back, F pour le front suivi d'un nombre.

Par exemple :

GEB-01, GEF-015

---

# CONCEPTION



# Maquettes

- Réalisées sous Photofiltre
- Définition d'une charte graphique  
(couleurs, forme des boutons, style des formulaires...)

CONNEXION

Email

Password

Se connecter



## SUSHI

Un sushi classique au saumon et à l'avocat  
entouré d'algue nori et composé uniquement  
de bonnes choses

3€

COMMANDER



# UML

Unified Modeling Language  
(Langage de Modélisation Unifié)

:

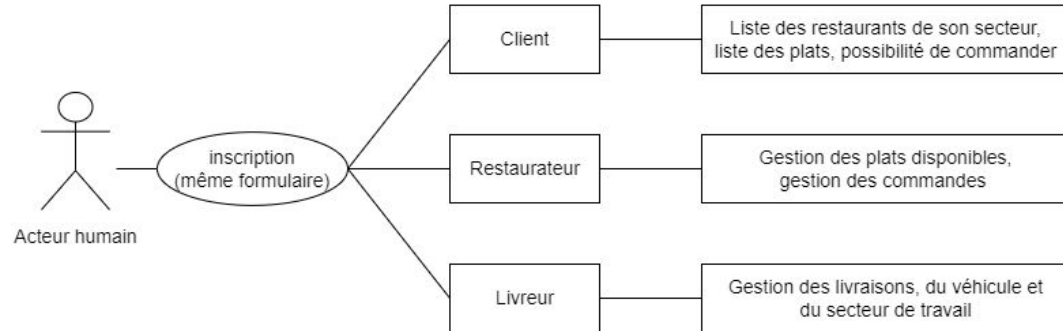
## → Drawio

Logiciel de schématisation en ligne

## → Cas d'utilisation

Sur l'exemple ci-joint : cas de l'inscription avec définition d'un rôle qui déterminera les fonctionnalités de l'utilisateur

## Inscription



# BASE DE DONNÉES

Données	Type	Taille	Genre	Id potentiel	Obligatoire
client_nom	alphanumerique	255	E	/	oui
client_prenom	alphanumerique	255	E	/	oui
client_id	numerique	255	E	oui	oui
client_rue	alphanumerique	255	E	/	oui
client_complement	alphanumerique	255	E	/	non
client_ville	alphanumerique	255	E	/	oui
client_telephone	numerique	255	E	/	oui
client_codePostal	numerique	5	E	/	oui
client_motdepasse	alphanumerique	255	E	/	oui
resto_raison_sociale	alphanumerique	255	E	/	oui
resto_numero_siret	alphanumerique	255	E	oui	oui
resto_adresse	alphanumerique	255	E	/	oui
resto_complement	alphanumerique	255	E	/	non
resto_ville	alphanumerique	255	E	/	oui
resto_telephone	numerique	255	E	/	oui
resto_type	alphanumerique	255	E	/	oui
resto_motdepasse	alphanumerique	255	E	/	oui
resto_codepostal	alphanumerique	255	E	/	oui
livreur_id	alphanumerique	255	E	oui	oui
livreur_nom	alphanumerique	255	E	/	oui
livreur_prenom	alphanumerique	255	E	/	oui
livreur_secteur	alphanumerique	255	E	/	oui
livreur_vehicule	alphanumerique	255	E	/	oui
livreur_telephone	numerique	10	E	/	oui
livreur_email	alphanumerique	255	E	/	oui
livreur_motdepasse	alphanumerique	255	E	/	oui
vehicule_id	alphanumerique	255	E	oui	oui
vehicule_type	alphanumerique	255	E	/	oui
commande_numero	alphanumerique	255	E	oui	oui
commande_rue	alphanumerique	255	E	/	oui
commande_complement	alphanumerique	255	E	/	non
commande_client	alphanumerique	255	E	/	oui
commande_resto	alphanumerique	255	E	/	oui
commande_montant	numerique	255	C	/	oui
commande_date	alphanumerique	255	E	/	oui
commande_horaire_livraison	alphanumerique	255	E	/	non
commande_codepostal	numerique	5	E	/	oui
commande_ville	alphanumerique	255	E	/	oui
commande_statut	alphanumerique	255	E	/	oui
statut_id	numerique	255	E	oui	oui
statut_nom	alphanumerique	255	E	/	oui

# Elaboration du dictionnaire de données

→ Référencement de toutes les données utiles au projet

→ Définition de l'ID de chaque table

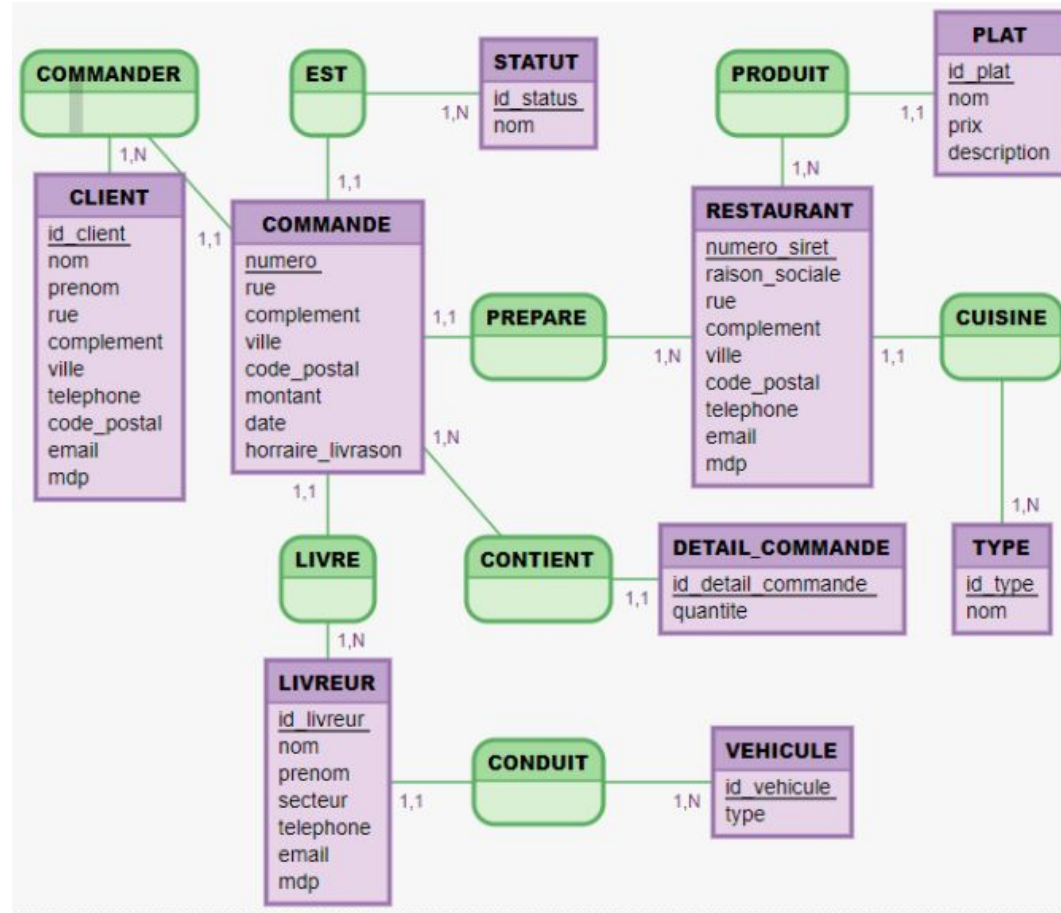
*Exemple ci-joint, dictionnaire de données complet dans le rapport écrit*

# MCD

→ Modèle Conceptuel de Données

→ Réalisé via Mocodo

→ Etablir les différentes relations entre les tables et schématiser la base de données du projet

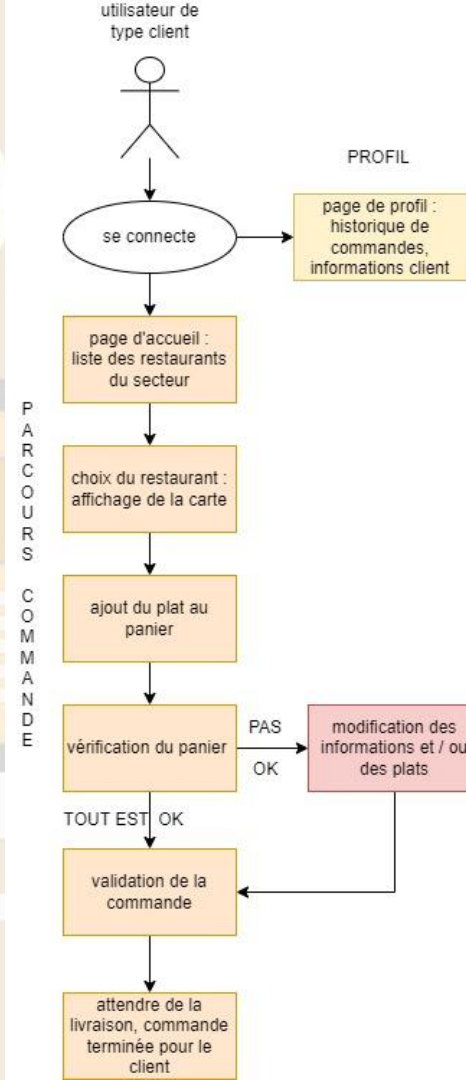


---

# ELABORATION DE L'APPLICATION

# Parcours client

Au départ, nous nous sommes particulièrement concentrés sur le parcours client, car nous voulions absolument terminer au moins un parcours dans son intégralité.



*UML pour le parcours commande d'un utilisateur de type client*

# ELABORATION DU FRONT



---

# HTML, Twig, CSS

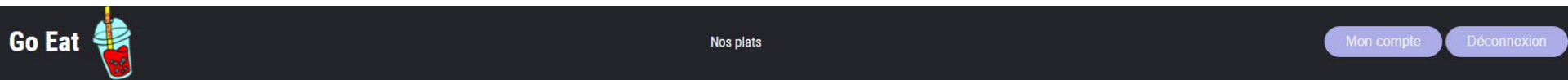
- Génération automatique du HTML grâce à Twig
  - Changement du HTML pour correspondre à notre identité visuelle
  - Style via du CSS en suivant les mêmes codes couleur et style général grâce à la normalisation des classes
-

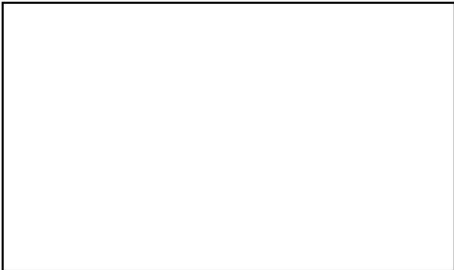
# Exemple pour la présentation des restaurants (page d'accueil client)

- **{% extends %}**  
permet de reprendre les paramètres du fichier de base notamment pour le style
- **{% for %}**  
boucle permettant de répéter le bloc de HTML pour tous les restaurants
- **{% endfor %}**

```
1 {% extends 'base.html.twig' %}
2
3 {% block title %}Go Eat
4 {% endblock %}
5
6 {% block body %}
7     <ul class="restaurateur">
8         {% for restaurant in restaurants %}
9             <li class="unRestau">
10                 <div class="vitrine"><img></div>
11                 <div class="detail">
12                     <span class="titleRestau">{{ restaurant.raisonSociale }}</span>
13                     <button>Afficher la carte</button>
14                 </div>
15             </li>
16         {% endfor %}
17     </ul>
18
19     <style>
20         .restaurateur {
21             width: 80%;
22             margin: 100px auto;
23             display: flex;
24             justify-content: space-evenly;
25         }
26         .unRestau {
27             list-style-type: none;
28             height: 400px;
29             width: 30%;
30             border: 2px solid black;
31         }
32         .vitrine {
33             width: 95%;
34             height: 65%;
35             margin: 2% auto;
```

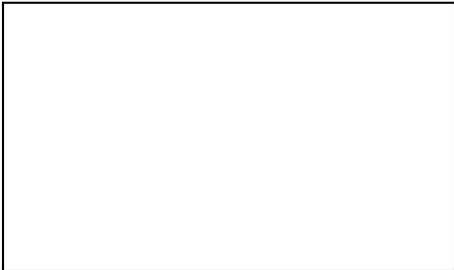
# Visuel de la présentation des restaurants





**Chaudron baveur**

Afficher la carte



**Horny Torinque**

Afficher la carte



**Chez momo**

Afficher la carte

---

**ELABORATION DU BACK**

# Exemple du controller pour les restaurants

(commandes en console)

→ Création d'une entité 'restaurant'

*symfony console make:entity* avec choix du nom et des propriétés de l'entité

→ Création du controller correspondant à l'entité avec création des différentes routes

*symfony console make:crud*

## Entité Restaurant

```
/**
 * @ORM\Entity(repositoryClass=RestaurantRepository::class)
 */
class Restaurant
{
    /**
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $raisonSociale;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $rue;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $ville;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $codePostal;
```

# Exemple du controller pour les restaurants

(modification du controller)

## → Modification des routes

Index : affichage de la page de compte d'un restaurateur

## → Modification du formulaire

Liaison du compte propriétaire à un restaurant

```

**
* @IsGranted("ROLE_RESTAURATEUR")
* @Route("/new", name="restaurant_new", methods={"GET", "POST"})
*/
public function new(Request $request, EntityManagerInterface $entityManager): Response
{
    $restaurant = new Restaurant();
    $villes = VilleApi::getVilles();
    $form = $this->createForm(RestaurantType::class, $restaurant);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $user = $this->getUser();
        $restaurant->setProprietaire($user);
        $entityManager->persist($restaurant);
        $entityManager->flush();

        return $this->redirectToRoute('restaurant_index', [], Response::HTTP_SEE_OTHER);
    }

    return $this->renderForm('restaurant/new.html.twig', [
        'restaurant' => $restaurant,
        'form' => $form,
        'villes' => $villes,
    ]);
}

```

Méthode d'ajout d'un restaurant par la route new

## Construction du formulaire restaurateur

```

<?php
danyrose, 3 days ago • entite,crud

namespace App\Form;

use App\Entity\Restaurant;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

danyrose, 3 days ago | 1 author (danyrose)
class RestaurantType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add('raisonSociale')
            ->add('rue')
            ->add('ville', null, [
                'attr' => ['list' => 'listeville']
            ])
            ->add('codePostal')
            ->add('complement')
            ->add('telephone')
            ->add('type')
        ;
    }

    public function configureOptions(OptionsResolver $resolver): void
    {
        $resolver->setDefaults([
            'data_class' => Restaurant::class,
        ]);
    }
}

```



# Avancée du projet

Application non terminée dans son intégralité

→ **Symfony**

Lacunes dans le langage, donc difficultés et temps de recherche important

→ **Premier projet**

Temps pour apprendre à connaître chaque membre du groupe et à définir comment travailler ensemble de manière efficace en ne laissant personne de côté ou en difficulté



# Conclusions personnelles

Malgré nos lacunes dans le langage puisque nous ne l'utilisons pas habituellement, nous restons satisfaits du résultat.

Nos différentes forces et faiblesses ont créé un groupe intéressant et efficace, basé sur le partage des connaissances.

L'organisation sur laquelle était basée le groupe nous a semblé correcte et fonctionnelle, mais surtout adaptée à nos différents profils.