Vaadin Flow Présentation

Applications multi-tiers

Edwin Häffner, Rachel Tranchida, Arthur Junod, Eva Ray

Table des matières

Projet: Pokémon TCGP builder

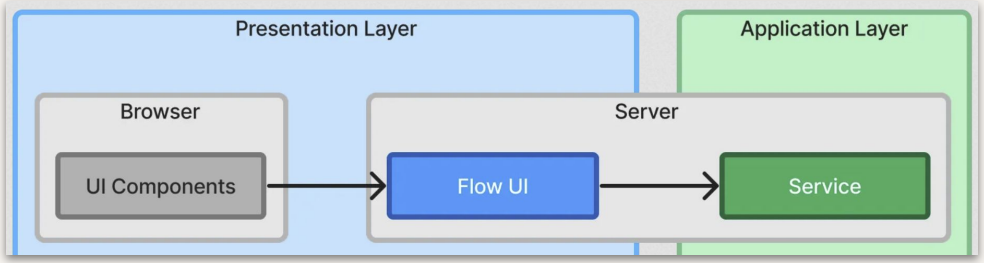
- **Pokémon TCGP**: Jeu vidéo mobile qui permet d'ouvrir des paquets de cartes Pokémon afin de constituer un deck et étoffer sa collection de cartes dans le but de prendre part à des combats.
- **Pokémon TCGP builder**: Application de construction de deck liée à Pokémon TCGP.
- **Fonctionnalités**: Authentification, Création de decks, Inventaire des cartes possédées, Parcours de tous les decks créés par tous les utilisateurs, Notation des decks, Notification par email quand un de nos decks est noté, Affichage des cartes manquantes pour la création du deck consulté...
- **Objectifs de l'application**:
  - Partage
  - Notation
  - Deck avec cartes possédées

Présentation de Vaadin Flow

- **Vaadin**: Framework Java, permet de créer un *frontend totalement en Java* en utilisant les composants Vaadin, sans avoir à écrire du code HTML/CSS/JavaScript
- **But**: Simplifier le développement d'interfaces utilisateur pour les développeurs Java, avec l'idée de créer des teams *full-stack*, plutôt que frontend et backend.
- **Caractéristiques principales**:
  - **Rendu côté serveur** (SSR): Le serveur envoie les màj de l'UI au client.
  - **Composants** accessibles, esthétiques et consistents prêts à l'emploi: Flex, Form, Notification, Login, EmailInput, ...
  - **Sécurité par défaut**: Vaadin sécurise et automatise la communication entre le navigateur et le serveur → clés de session automatisées, validation côté serveur et vérification des entrées.

Fonctionnement de Vaadin Flow

- **Initialisation du client (bootstrapping)**: Le navigateur charge une page HTML avec des *éléments statiques* et le *moteur de rendu client-side* de Vaadin. Le serveur envoie des *instructions de rendu sous forme de JSON* pour construire ou update le DOM.
- **Evénements**: Les actions utilisateur (ex. clics) sont envoyées au serveur via HTTP POST. Le serveur exécute les listeners Java associés.
- **Mise à jour de l'UI**: Les modifications côté serveur génèrent un nouvel objet *JSON contenant des instructions de mise à jour*. Le moteur de rendu client met à jour le DOM en conséquence.
- **Navigation**: Les clics sur les liens sont *interceptés* par Flow. Flow remplace *dynamiquement* la vue côté serveur et renvoie les *instructions* JSON.



Construire une interface

- Tout est un composant
- On utilise des layouts pour construire les vues
- On utilise des événements pour les interactions

```
new Grid<Employee>()
```

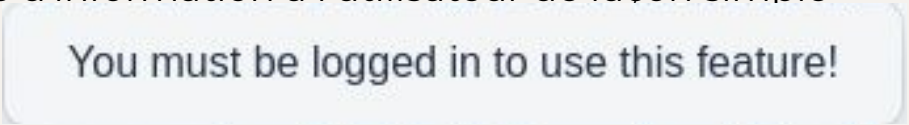
```
new VerticalLayout(
      new Button("Button 1"),
      new Button("Button 2")
);
```

```
var button = new Button("Greet")

button.addClickListener( e -> {
    add("Welcome!")
})
```

Implémentation de Vaadin

- Utilisation de vues (Annotation `@Route(name)`)
- En général la vue en elle même est un layout (`extends VerticalLayout`)
- Fonctions supplémentaire ->
  - `implements KeyNotifier` pour écouter le clavier
  - `implements BeforeEnterObserver` pour exécuter fonctions avant d'arriver sur la page
  - `implements AfterNavigationObserver` pour exécuter des fonctions après
- Formulaires
  - `TextField, PasswordField, EmailField`
- Notifications -> Permet l'affichage d'information à l'utilisateur de façon simple



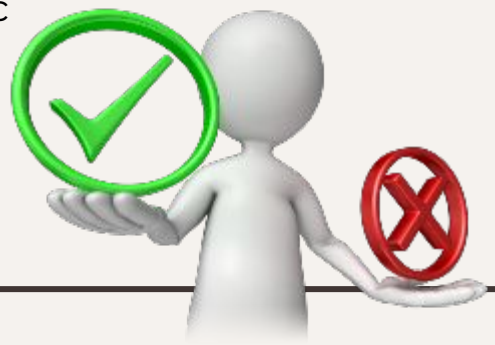
Points forts… et points faibles

Points forts

- **Facilité** de développer l'UI
- Beaucoup de commandes à disposition
- Routage simplifié
- Pas besoin de gérer une API REST de façon explicite (Annotations)
- Backend et frontend moins séparé, **même langage** sur l'ensemble du projet (JAVA)
- Fonctionne sur **différents frameworks**…

Points faibles

- Pas imperméable à la **notion de CSS**, certaines fonction n'existent pas.
- Certaines fonction nécessitent l'**utilisation de Spring Boot**
- **Moins libre** dans la création de l'ui
- **Vaadin est propriétaire** et pas 100% gratuit.
- **Endpoints** avec Quarkus



Feedback : Utilisation de Java

- **Facilité d'utilisation :** Facile à prendre en main et comprendre. Pour nous qui avons une bonne expérience de développement en java, la courbe d'apprentissage est raisonnable
- **Organisation en classe et héritage**
- **Factorisation instinctive**

```java
// Classe qui affiche une carte simple avec des boutons et un compteur
public class CardWithCounter extends CardBasic implements KeyNotifier {

    protected int selectedCount;
    protected final TextField selectedCountField;
    // Counter components
    protected Button minusButton;
    protected Button plusButton;
}
```

```java
// Classe pour afficher une carte simple avec son image
public class CardBasic extends VerticalLayout {

    protected final CardDTO card;

    protected Image cardImage;

    private static Image getImage(CardDTO card) {...}
```

Feedback : Mise en place du layout

- Possibilité de créer facilement des layouts et les ajouter
- Récupérer le style et le modifier (les attributs CSS ne sont pas toujours re-définis par une fonction java)
- Facile de combiner les éléments

```java
@PostConstruct
private void init() {

    MenuBar menuBar = new MenuBar();

    // Create a flex layout container for the decks
    FlexLayout cardContainer = new FlexLayout();
    cardContainer.setWidth("100%");
    cardContainer.getStyle()
            .setDisplay(Style.Display.FLEX)
            .setFlexWrap(Style.FlexWrap.WRAP)
            .setMargin("16px")

.setJustifyContent(Style.JustifyContent.FLEX_START)
            .set("gap", "16px");
      …
add(menuBar, cardContainer);
```

Feedback : Utilisation directe des services

- On peut injecter directement les services dans nos views
- Dans un @PostConstruct pour s'assurer de leur injection, on peut récupérer les informations nécessaires de la DB

```
public class AllDeckView extends VerticalLayout {


    @Inject
    DeckService deckService;

@PostConstruct
private void init() {

   //…


    // Get cards from database
    List<DeckDTO> decksDTO =
deckService.getDecksOrderedByRating();

}
```

Feedback : Utilisation du clavier

- Il suffit d'ajouter un listener à la classe implémentant un keyNotifier
- Facile d'associer des touches à une action
- Point négatif : Pas toutes les touches sont disponibles

```
public void setKeyboardShortcuts(KeyNotifier
keyNotifier) {
   keyNotifier.addKeyDownListener(Key.ARROW_RIGHT,
e -> this.handleRightArrow());
   keyNotifier.addKeyDownListener(Key.ARROW_LEFT,
e -> this.handleLeftArrow());
   keyNotifier.addKeyDownListener(Key.ARROW_UP, e
-> this.handleUpArrow());
   keyNotifier.addKeyDownListener(Key.ARROW_DOWN,
e -> this.handleDownArrow());
   keyNotifier.addKeyDownListener(Key.INSERT, e ->
this.handleAddCard());
   keyNotifier.addKeyDownListener(Key.DELETE, e ->
this.handleRemoveCard());

}
```

Feedback : Authentification

- **Gros problème : Implémentation de l'authentification à l'aide d'annotation**

1. Utilisation de `quarkus security jpa` impossible à cause de la manière duquel vaadin organise REST.

2. Utilisation des annotations de Vaadin impossible à cause de l'utilisation de quarkus.

Solution : Gérer l'authentification à la main

```
public class AuthorizedView extends
VerticalLayout implements
BeforeEnterObserver {

    @Override
    public void beforeEnter
        (BeforeEnterEvent event ) {
        if (!UserHelper.isLoggedIn()) {
            VaadinSession
            .getCurrent().setAttribute(
                LOGIN NOTIFICATION KEY, true);
            event.rerouteTo(MainView.class);}
        }
    }
}
```

Feedback : Authentification (Suite)

Au moins, Vaadin nous permet d'utiliser des sessions.

```
public class UserHelper {
    //Getters for id, username and DTO

    public static void setCurrentUser(UserDTO user) {
        setCurrentId(user.id());
        setCurrentUsername(user.username());
        VaadinSession.getCurrent().setAttribute("userDTO", user);
    }

    public static boolean isLoggedIn() {
        return getCurrentId() != -1;
            //Returns -1 if user isn't logged in }

    public static void logout(){
    VaadinSession.getCurrent().setAttribute("userId", -1);
    VaadinSession.getCurrent().setAttribute("user", null);
    VaadinSession.getCurrent().setAttribute("userDTO", null);
    }
}
```

Vaadin utilise un cookie avec un sessionId (nommé `JSESSIONID`)

-> Permet d'associer des informations avec une session

Questions?