

 Display-Interactive	<b>DI-UGO-TEST-FULL-STACK-1-0</b> Test Full Stack	Revision: 1.0 Issued: 2020-11-09
--	--	-------------------------------------

## Définition du test

Le but de ce test est de tester vos compétences sur PHP/Symfony d'un côté et sur ReactJS / Typescript de l'autre.

### Backend

Sur le backend vous devez utiliser le framework Symfony 5.x / 6.x. Vous devez utiliser une BDD sqlite (pour faciliter le transport).

Un bonus est accordé si vous avez au moins un test unitaire sur le controller ou sur la commande Symfony

### Frontend

Vous devez utiliser React.

Vous devez utiliser TypeScript (<https://www.typescriptlang.org/>) et non du simple Javascript.

Vous pouvez utiliser le framework CSS que vous voulez. Vous êtes libre du layout. Vous ne serez pas jugé sur l'aspect design. Laissez libre court à votre créativité !

Un gros bonus est accordé si au moins un composant React est testé avec un framework de test.


## Rendu du test

Vous devrez fournir un lien github ou récupérer le zip contenant tous les éléments permettant d'installer et de visualiser votre projet.

Avec dans le zip deux répertoire

- **backend** : l'application symfony. prête à être installée via composer install
- **frontend** : l'application ReactJS, prête à être installée via npm i, et lancée via npm start

Un README.md devra être fourni avec la méthode de build et de test (si des tests ont été écrit)

 Display-Interactive	<b>DI-UGO-TEST-FULL-STACK-1-0</b> Test Full Stack	Revision: 1.0 Issued: 2020-11-09
--	--	-------------------------------------

## Définition de l'application

Vous allez devoir créer une application qui permet de voir la liste des clients d'une société ainsi que l'ensemble des commandes pour chaque client.

Pour se faire vous allez devoir importer une liste de clients (customers) et de commandes (orders) dans une BDD Sqlite. Exposer ses données via une API, et afficher ces données dans une application React.

### Backend

Vous devrez effectuer votre développement en PHP, en Symfony 4.0x. Vous devrez créer une application Symfony vierge avec vos développements.

Vous devez créer une commande Symfony **ugo:orders:import** qui va lire 2 fichiers csv fournis

- "customers.csv"
  - Le numéro client est unique
  - Mapping pour la civilité (Fichier -> API), colonne "title" dans le CSV :
    - 1 -> mme
    - 2 -> m
- "purchases.csv"
  - Un client peut avoir 0..n achats

Remplir une base de donnée sqlite (plus facile à transporter) avec deux Entités

- Customer
- Order (many to none)

Créer l'API suivante :

- /customers = la liste de tous les customers au format json
- /customers/<customer\_id>/orders = la liste des orders pour le customer avec l'id = customer\_id au format json

 Display-Interactive	<b>DI-UGO-TEST-FULL-STACK-1-0</b> Test Full Stack	Revision: 1.0 Issued: 2020-11-09
--	--	-------------------------------------

## Frontend

En utilisant l'api que vous avez crée côté backend, créez une application react qui affiche 2 Pages

- La liste des customers
- La liste des orders pour un utilisateur

### Page Customers

Elle est constitué d'un tableau contenant les champs

- id du user
- title
- lastname
- firstname
- postal\_code
- city
- email
- show orders

Sur chaque ligne **show orders** est clickable et permet d'accéder à la page Orders

### Page Orders

Elle est constitué

D'un lien Back permettant de revenir à la liste des customers

D'un tableau contenant les champs

- last\_name : le last name du client
- purchase\_identifier
- product\_id
- quantity
- price
- currency
- date

En bas du tableau on retrouve un champ total : avec le total du prix cumulé de tous les orders