RAPPORT DE STAGE

Création d'une application mobile :

Du 4 janvier au 26 février 2021

Lecoublet Théo

Deuxième années – Spécialité SLAM

ÉTABLISSEMENT SAINT-ADJUTOR

BTS SIO

Etablissement Saint-Adjutor | BTS Services Informatiques aux Organisations

CIAGE SYSTÈMES INTERNET MOBILITÉ

Sommaire

Remerciements

Dans un premier temps, je tiens à remercier Monsieur Nordine Kecita, gérant de l'entreprise CIAGE, m'ayant permis d'intégrer en tant que stagiaire durant deux mois cette société.

Dans un second, remercier Adel Bentoumi, développeur web angularjs, grâce à qui j'ai pu comprendre la base existante de l'application.

Et enfin un grand merci à Guillaume Bentoumi, informaticien réseau, qui m'a accompagné et aidé de l'installation et la compréhension de l'existant à la réflexion et développement de l'application.

Je les remercie à la fois de m'avoir gentiment accueilli, permis d'apprendre davantage, et accompagné avec leur bonne humeur et leur sympathie.

Présentation Entreprise

Simuleo est une marque de la société CIAGE existant depuis 1997, cette dernière étant à l'origine essentiellement une société de création de logiciels métiers sur mesure. Par la suite, CIAGE se tourne progressivement vers les technologies embarquées. La mobilité devenant un axe de développement stratégique. En plus de son activité d'agence digitale, CIAGE acquiert une expertise en développement sur plateformes mobiles. Ainsi en 2013, CIAGE se voit créer Simuleo. De nos jours CIAGE se concentre sur le maintien et l'évolution constante de sa marque Simuleo

Photo façade

https://www.google.com/maps/@48.9946857,1.6947406,17.25z?hl=fr

Présentation du service d'accueil et des moyens informatiques

Ayant travaillé en télétravail, je n’ai pas pu physiquement voir le matériel et moyen informatique de la société.

Cependant, durant mon stage, l’entreprise était en pleine migration de serveur, leur donnant momentanément deux serveurs existant en tant que moyen informatique.

De plus, les employés, comme ici Guillaume Bredzinski, était équipé d’un mac mini lui permettant de travailler sous cet os. Ainsi, pour par exemple mettre en application sous Ios cela est possible. En effet sans un appareil sans ios, impossible d’exporter et mettre en place l’application pour ios sur l’AppStore.

Plus généralement, pour ce qui est des moyens logiciel, les employés travaillent sous des appareils fonctionnant sous Linux ou Ios.

Présentation du sujet

Description :

Simuleo est un simulateur de menuiserie. C'est-à-dire, un logiciel qui permet d'importer la photo son habitat et de superposer des produits de menuiseries comme des portes ou des fenêtres, permettant d'avoir une projection de la finalité d'un chantier directement sur un écran.

Motivations :

Simuleo permet de présenter plus clairement des catalogues jusqu'alors complexe, avec une esthétique pouvant être spécifique à chaque demandeur, aidant ainsi le particulier ciblé quant à ses choix vis-à-vis de la finalité du projet.

Objectif du demandeur :

Les demandeurs sont essentiellement des professionnels, entre fabricants de menuiseries ou des revendeurs. Cette application leur permet d'accélérer la démarche de vente, trouver de nouveaux client potentiel à travers le simulateur.

Développement

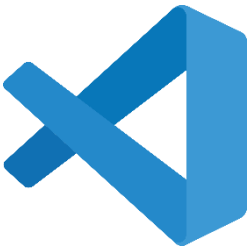
Outils

Environnement

Développement sous le système Debian : KDE Plasma

Terminal emulator: Kitty

Framework: ohmyzsh



IDE : VisualStudioCode

Emulateur smartphone : Android-Studio

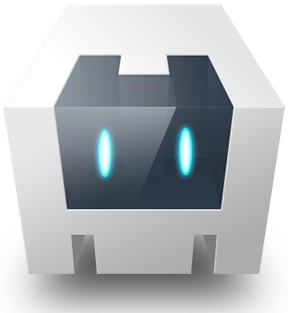
Gestion version : GitHub

Framework et platform

Ionic 5

Angular

Nodejs



Cordova

NPM gestionnaire de paquets de Node

NVM gestionnaire de version Node

Démarche

Description de la démarche :

Pour la démarche établie, il a d'abord était question d'analyser le projet à créer, celui existant et leur congénère web.

Par la suite, grâce à une analyse des différences entre application déprécié et application web, on établir les fonctionnalités supplémentaires à ajouter, tout en prenant on compte le fait que c'est un projet pour mobile.

Enfin la réalisation, mettant en œuvre les différents aspects vu auparavant dans une interface utilisateur fonctionnelle, communiquant avec une ApiRest.

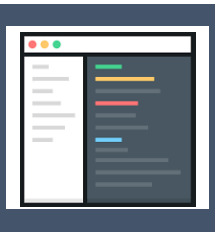
Schéma de la démarche :

Analyse du projet

- Une version mobile (dépréciée) existe deja
- Ainsi qu'une version web
- Analyse des technologies

Définition des fonctionnalités

- Après analyse, on connait les fonctionnalités à ajouter
- Prise en compte de l'aspect mobile pour l'experience utilisateur



Réalisation

- Mise en place/ Adaptation de la communication avec l'api
- Programmation de l'interface

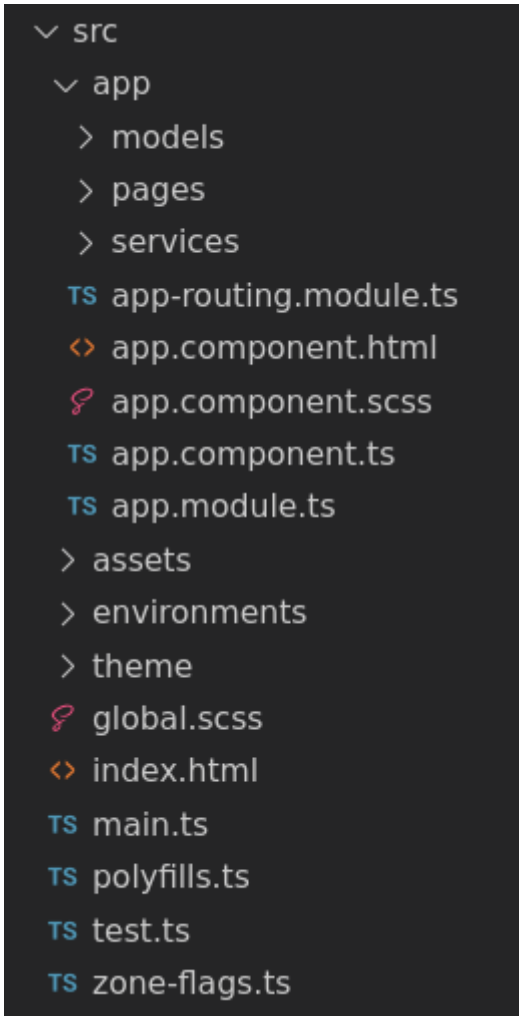
Choix effectués

-A l'origine, je ne devais que simplement mettre à jour une application. Cependant, après une semaine d'installation compliquée, il est apparu que maintenir une application aussi déprécié était impossible, de plus que aucun commentaire ni documentation vis-à-vis de l'application mobile n'existait. Ainsi il a été décidé de refaire l'application avec les dernières versions des outils utilisés lors de la première application.

-Une interface sobre, avec le strict minimum pour que l'expérience sur l'application mobile corresponde à celle de l'application web. Ainsi qu'une mise en svg des icons utiliser par les applications.

Détails technique :

Architecture du projet :

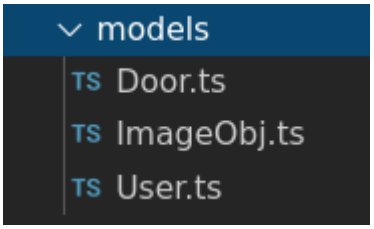


Modèles de données :

User : gérer l'utilisateur et ses données une fois identifié

ImageObj : Pour chaque objet/image déplaçable, ceci permet d'y renseigner diverses données

Door : Une porte contient diverses données telles que la forme, la couleur etc. Ainsi que deux ImageObj

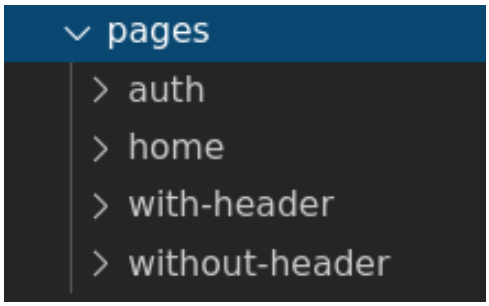


```
export class Door  {
    public code:string;

    public image:ImageObj;
    public background:ImageObj;

    public forme;
    public gamme;
    public modele;
    public cadre;
    public vitrage;
    public couleur;
    public poignee;
    public finition;
    public matiere;

    public selectedPart;
```

```
export class User  {

    public uuid: string;
    public slug: string;
    public dataProfile: string;

    constructor(uuid: string, slug: string){
        this.uuid = uuid;
        this.slug = slug;
    }
```

```
export class ImageObj  {
    public link;
    public coordX;
    public coordY;
    public size; //height
    public rotate;

    constructor(link:string){
        this.link = link;
```

Pages :



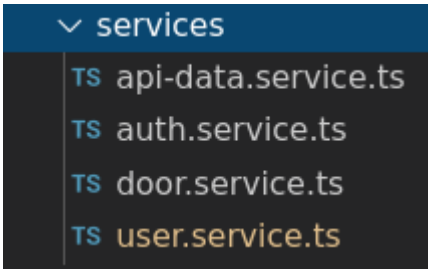
auth : gérer/afficher la page d'authentification

home : gérer/afficher la page principale continuellement chargé une fois connecté

with/without-header : gérer/afficher la page de contenu de la sélection en fonction de valeurs reçues

Services :

Généralement une instance de classe qui permet de factoriser certaines fonctionnalités ou d'accéder à un état permettant ainsi aux composants de communiquer entre eux.



api-data : gère les données avec des appels api

auth : gère l'authentification liant auth.page et user.service

door /user: mutateur et accesseur de leur model de données respectif

Programmation:

Une grande partie du projet repose sur la communication avec l'api Simuleo. En effet le service api-data joue le rôle de traducteur. Il récupère ou envoie des fichiers json, et les trie.

Par exemple, pour la partie d'acquisition du json d'une porte sous cette forme :


```
"matieres": [ …
],
"gammes": [ …
],
"formes": [ …
],
"collections": [ …
],
"elements": [
    {
        "id": 1,
        "name": "Porte",
        "panneaux": [ …
        ],
        "couleurs_panneau": [ …
        ],
        "batis": [ …
        ],
        "couleurs_bati": [ …
        ],
        "vitrages": [ …
        ],
        "poignees": [ …
        ],
        "accessoires": [ …
        ]
    }
],
"gammes_collections": [ …
],
"gammes_matieres": [ …
],
"style": [ …
],
"vitrage_types": [ …
],
"couleur_group": [ …
]
```

Dans le service, et pour chaque élément de ma porte on trie le json avec ces accesseurs :

```
getMatieres(menu){…
}

getGammes(menu){…
}

getFormes(menu){…
}

getModeles(menu){ //Aussi nommé Collection/ panneau…
}

getCadre(menu){ // Aussi nommé Batis…
}

getPoignees(menu) {…
}

getVitrage(menu) { // THERE IS A PRBLM …
}

getCouleur(menu) { // Panneau + batis | header : couleur_group (option)   && Mettre texte en dessou
}
// OPTIONS ()

getAccessoires(menu) { // Aussi nommé Finition …
}


getPetitBois(menu){…
}

getPetitBoisCouleur(menu){ …
}
```

Ainsi on se retrouve avec au total plusieurs json normés pouvant être dynamiquement géré dans les pages de contenue (**with/without-header.page**) :

With-header.page*

```
<ion-content>
  <ion-header>
    <ion-toolbar>
      <ion-title>{{selectedPage}} </ion-title>
    </ion-toolbar>
    <ion-segment scrollable value="heart">
      <div *ngFor="let listObject of selectedObject">
        <ion-button size="small"
                    (click)="onSelectObject(listObject.key)"
                    [ngClass]="{'button': true,
                                'active': selectedObjectName == listObject.key}" >
          {{ listObject.key}}
        </ion-button>
      </div>
    </ion-segment>
  </ion-header>


  <div>
    <app-content [object]="givenList" (onBackToHome)="onBackToHome()"></app-content>
  </div>


</ion-content>
```

```
<div class="img-list">
  <div *ngFor="let oneObject of object" >
    <img src="{{oneObject.img}} " alt="Si cette ima
         (click)="changeObject(oneObject.refp)">
  </div>
</div>
```


JSON RENDER FOREACH PARAM OF DOOR CLASS

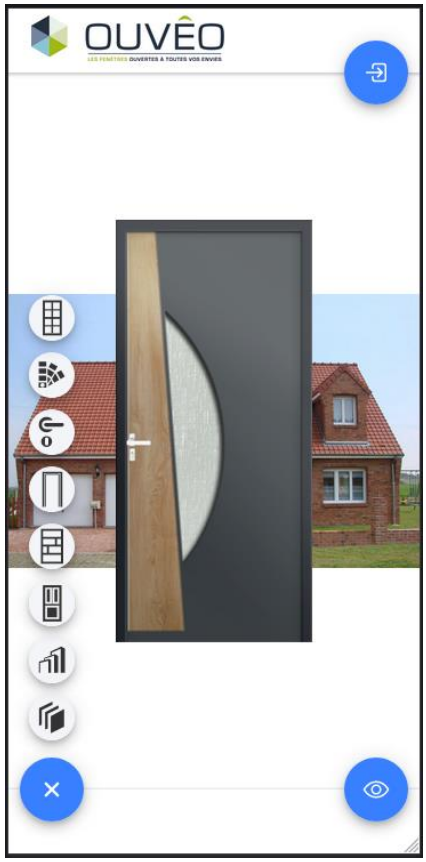
```
{
  "Gamme":[
    {
      "id": 1,
      "name": "Contemporain",
      "refp": "CONTEMPORAIN",
      "displayName": "Contemporain",
      "gamme": [
        {
          "id": 1,
          "name": "Évolution",
          "refp": "EVOLUTION",
          "displayName": "Évolution",
          "id_style": 1,
          "img": "https://api.simuleo.com/data/ouveo-20200909/file/thumbnails/...atiere-1-1.png?
uuid=c5596b7a-a3f8-4884-abc0-be3d28d86d11"
        },
        {
```

Maquette:

Connexion

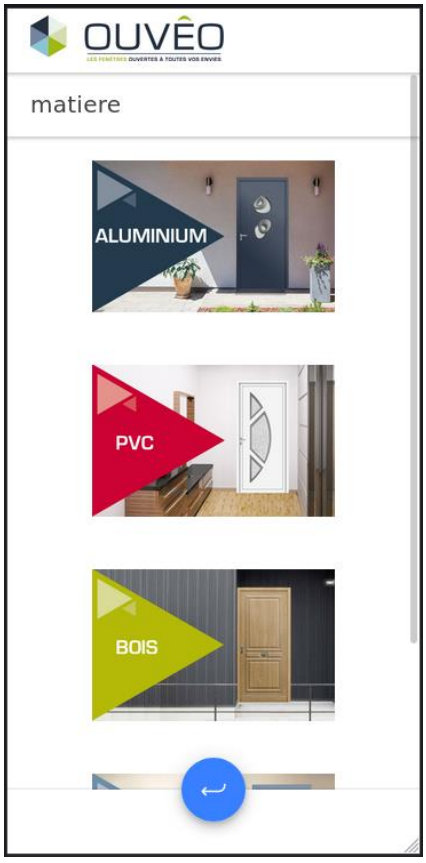
Si plusieurs profiles

Home(image movable)



Avec entête

Sans entête



Bilan

Selon moi, ce stage m'a apporté beaucoup de connaissance.

Dans un premier temps, cela m'a plongé pendant deux mois non-stop, sous Linux (chose que j'apprécie), et donc par conséquent apporté plus de connaissances vis-à-vis de l'os.

Dans un second, le développement mobile m'a appris à concevoir une application d'une autre manière que les applications web classique. Utilisation de composant native par exemple (camera, fichier stockés, etc).

Mais aussi utilisation d'un nouveau framework, enfin deux nouveaux. Même si apprendre deux nouvelles technologies et produire une solution complète et fonctionnelle en 2 mois est une tâche qui n'est pas des plus faciles, cela reste très enrichissant. Mettant en avant les problèmes de respect de délais que les entreprises font en sorte d'éviter.

Enfin cela m'a permis de m'intéresser d'avantage au fonctionnement et principe d'ApiRest. Des normes nécessaires pour obtenir l'appellation Restfull d'une Api. Messages de retour (200,400,500,..), les normes de lecture et d'envois de json, etc.