Université Abdelmalek Essaâdi

Faculté Polydisciplinaire – Larache

Programmation Full Stack

Dev Info et Méthodes DevOps

Pr : Kouissi Mohamed

Rapport Du Projet

Réalisé par : Mouad Mounad SMI0155/23

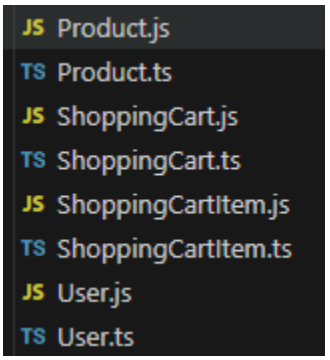
❖ Idée Générale

Ce rapport a pour objectif d'appliquer les concepts fondamentaux du développement web avec le Framework **Angular** .

➢ La base de ce projet : Type Script (models)

- se compose de 4 fichier de type : typescript

Nous avons développé une petite application en TypeScript en appliquant les fondamentaux de la programmation orientée objet **.**
L'objectif était de comprendre comment créer des **classes et constructeurs**, gérer des **attributs et méthodes** , et en fin la création des **objets .**



Code et affichage :

User

```
let user: User = new User("001");
user.lastName = "Mounad"
user.firstName = "Mouad"
user.userType = userType.Admin
console.log(user.greetUser())
```

```
Admin
Hello Mounad Mouad
```

Product

```
let product = new Product("001")
product.productTitle = "Samsung ultra 25"
product.productPrice = "15000 DH"
console.log(product.printProduct())
```

```
productID:001, productTitle: Samsung ultra 25, productPrice: 15000 DH
```

Shoppingcart / Shoppingcart

```
var productFour = new Product_1.Product("004");
productFour.productTitle = "TV 50 Pouce";
productFour.productPrice = "6000 DH";
var productFive = new Product_1.Product("005");
productFive.productTitle = "IPHONE 14 PRO";
productFive.productPrice = "13000 DH";
```

```
public addItem(shoppingCartItem: ShoppingCartItem){
    let elem : ShoppingCartItem | undefined = this.itemsProduct.find(
        x => x.itemProduct.productID == shoppingCartItem.itemProduct.productID)
    if(elem == undefined){
        this.itemsProduct.push(shoppingCartItem)
    } else {
        elem.addProduct(shoppingCartItem)
    }
}
```

```
public addProduct(shoppingCartItem: ShoppingCartItem) {
    if (this.itemProduct.productID == shoppingCartItem.itemProduct.productID) {
        this.quantity += shoppingCartItem.quantity
    }
}
```

```
shoppingCart.addItem(shoppingCartItemFive);
```

```
public getItems(){
 for(var i in this.itemsProduct){
    console.log(this.itemsProduct[i].displayProduct() +"\n");
        }
    }
}
```

```
public displayProduct() {
    return "Title: " + this.itemProduct.productTitle + ", quantity: " + this.quantity
}
```

```
console.log(shoppingCart.getItems());
```

```
Title: IPHONE 14 PRO, quantity: 1
```

```
public subtractProduct(shoppingCartItem: ShoppingCartItem) {
    if (this.itemProduct.productID == shoppingCartItem.itemProduct.productID) {
        this.quantity -= shoppingCartItem.quantity
    }
}
```

```
    public removeItem(shoppingCartItem: ShoppingCartItem){
let elem : ShoppingCartItem | undefined = this.itemsProduct.find(x => x.itemProduct.productID == shoppingCartItem.itemProduct.productID)
if(elem != undefined){
    elem.subtractProduct(shoppingCartItem)
    if(elem.quantity == 0){
 this.itemsProduct.splice(this.itemsProduct.indexOf(shoppingCartItem), 1)
        }
    }
  }
```

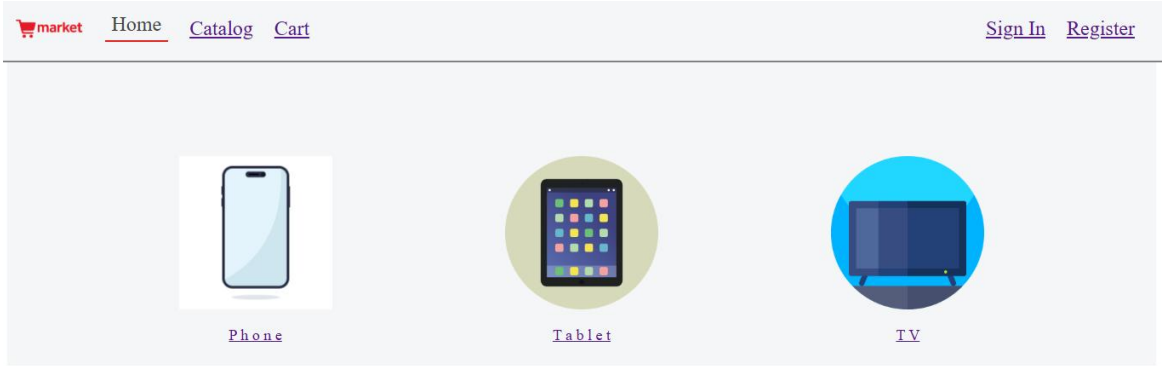
```
shoppingCart.removeItem(shoppingCartItemFive);
```

❖ Projet Final :

➢ Introduction

Ce projet consiste à développer une application E-commerce qui gère l'affichage des produits par catégories , et aussi l'achat de ces produits .

L'interface d'application est comme suit :

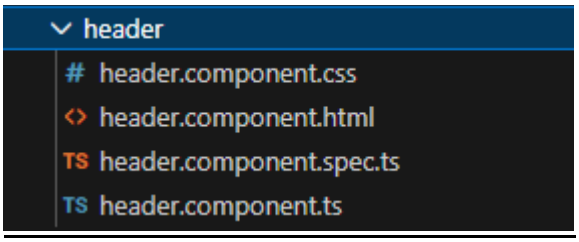


Les parties de cette UI sont : Home , Catalogue , Cart et aussi le sign in .

➢ Corps

1. 1er Composant est le Header qui contient :

La navigation << Home, Catalog, Cart >> et les liens d'authentification << Sign In, Register >> .



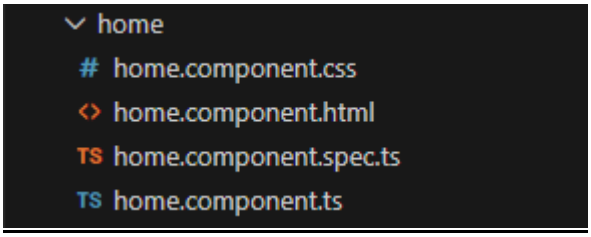
Header-Html

```
<img class="logo" src="/assets/images/marketpng.png" alt="Logo" />
<a routerLink="/home" routerLinkActive="active" >Home</a>
<a routerLink="/catalog" routerLinkActive="active">Catalog</a>
<div class="cart">
  <a routerLink="/cart" routerLinkActive="active">Cart</a>
</div>
```

```
<a href="" class="reg">Register</a>
```

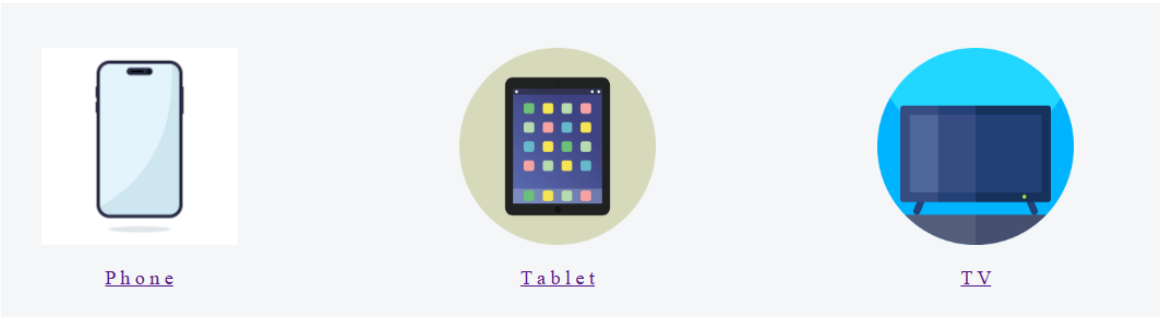
2. **2eme Composant est le Home qui contient :**

Présentation des 3 catégories << Phone , Tablet , Tv >> , en plus les accès rapide vers le composant catalogue .



Home-Html

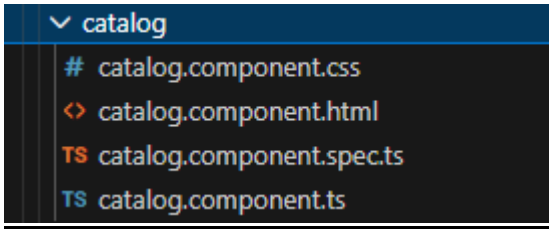
```
<li>
  <a routerLink="/catalog" [queryParams]="{filter:'phone'}" class="part">
    <img src="assets\images\mobile.avif" alt="Smart Phone" />
    <div>P h o n e</div>
  </a>
</li>
<li>
  <a routerLink="/catalog" [queryParams]="{filter:'tablet'}" class="part">
    <img src="/assets/images/tablet.png" alt="Tablet" />
    <div>T a b l e t</div>
  </a>
</li>
<li>
  <a routerLink="/catalog" [queryParams]="{filter:'smarttv'}" class="part">
    <img src="/assets/images/tv.png" alt="Smart TV" />
    <div>T V</div>
  </a>
</li>
```



---

3. **3eme Composant est le Catalog qui contient :**

L'affichage de la liste des produits d'après leurs catégorie (phone, tablet, tv) .



Catalog-Html

```
<a routerLink="/catalog" [queryParams]="{filter:'phone'}" routerLinkActive="active" class="button" i18n> Smart Phone </a>

<a routerLink="/catalog" [queryParams]="{filter:'tablet'}" routerLinkActive="active" class="button" i18n> Tablet </a>

<a routerLink="/catalog" [queryParams]="{filter:'smarttv'}" routerLinkActive="active" class="button" i18n> Smart TV </a>

<a routerLink="/catalog"  routerLinkActive="active" [routerLinkActiveOptions]="{exact:true}" class="button" i18n> All </a>
```

```
<ul class="products">
  <li class="product-item" *ngFor ="let _product of getFilteredProducts()" >
   <app-product-details [product]="_product"

                (buy)="addToCart(_product)" ></app-product-details>
  </li>
</ul>
```

Catalog-Ts (En utilisent les services)

- **L'utilisation de Cart Service :**

```
addToCart(product: Product){
  this.cartService.addToCart(product)
  this.router.navigate(['/cart'])
}
```

```
getFilteredProducts() {
  return this.filter === ''? this.products : this.products.filter(
    (product: any) => product.category === this.filter
  );
}
```

Smart Phone | Tablet | Smart TV | All

- 
  001
  IPhone 17
  Part Type: IPhone 17
  17000 DH
  Buy
- 
  002
  Samsung Ultra
  Part Type: Samsung Ultra
  15000 DH
  Buy

Smart Phone | Tablet | Smart TV | All

- 

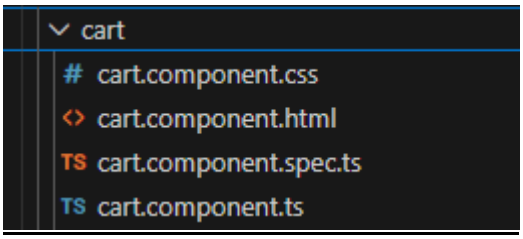
  001
  Samsung Tab
  Part Type: Samsung Tab
  2400 DH
  Buy
- 
  002
  IPad
  Part Type: IPad
  3500 DH
  Buy

Smart Phone | Tablet | Smart TV | All

- 
  001
  Smart TV 48 Pouce
  Part Type: Smart TV 48 Pouce
  9500 DH
  Buy
- 
  002
  LG Smart TV 48 Pouce
  Part Type: LG Smart TV 48 Pouce
  18000 DH
  Buy

4. **4eme Composant est Cart qui contient :**

La gestion du panier qui peut afficher les produits ajoutés et aussi la suppression du produits .



Cart-Html

```
  <ul class="cart" *ngIf="cartItems.length > 0">
      <li class="cart-item" *ngFor="let product of cartItems">
        <div class="product">
<div class="product-details">
  <img [src]="getImageUrl(product)" [alt]="product.productTitle" />
  <div class="product-info">
    <div class="name">{{ product.productTitle }}</div>
    <div class="description">{{ product.productTitle }}</div>
    <div class="category">Part Type: {{ product.category }}</div>
  </div>
</div>
<div class="price">
  <div> {{ product.productPrice }} </div>
```

```
<button (click)="removeFromCart(product)">Remove</button>
```

Cart-Ts (En utilisent les services)

```ts
getImageUrl(product: Product){
  return '/assets/images/'+ product.productImage
}
```

```ts
removeFromCart(product: Product) {
  this.cartService.remove(product);
}
```

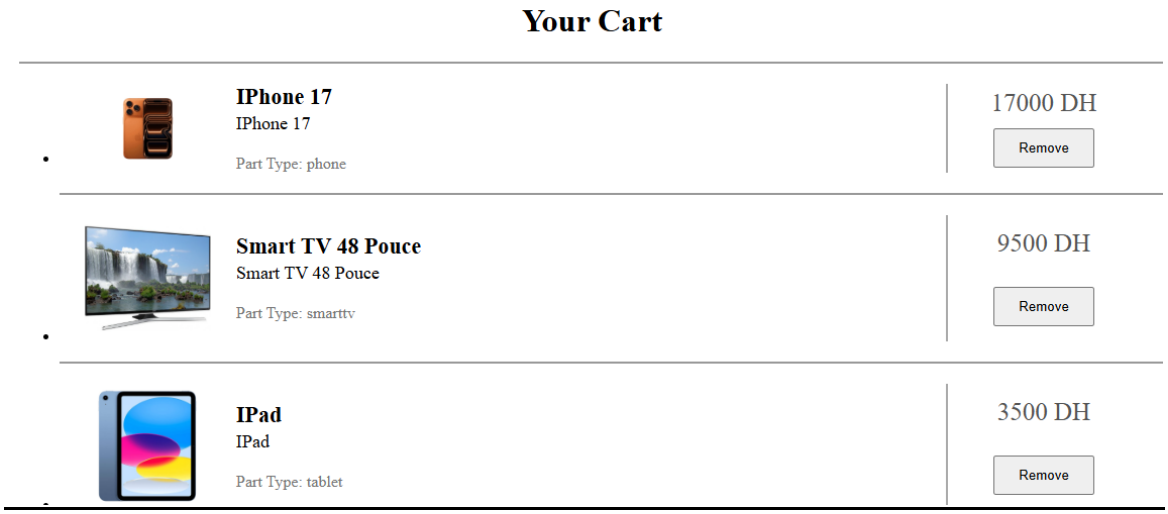
```ts
private baseUrl = 'http://localhost:3000/api';

cart: Product[] = []
```

```ts
constructor(private http: HttpClient) {
  this.http.get<Product[]>(`${this.baseUrl}/cart`).subscribe(cart => {
    this.cart = cart
  })
}
```

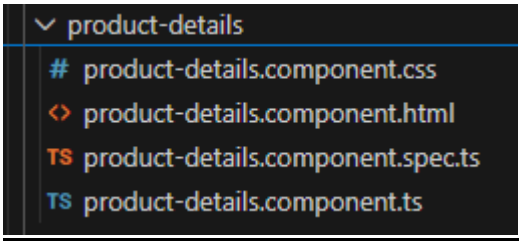
```ts
addToCart(product: Product){
  this.cart.push(product)
  this.http.post(`${this.baseUrl}/cart`, this.cart).subscribe(() => {
    console.log("product :" + product.productTitle +" added to cart")
  })
}
```

```ts
remove(product: Product) {
  let newCart = this.cart.filter((i) => i !== product);
  this.cart = newCart;
  this.http.post(`${this.baseUrl}/cart`, this.cart).subscribe(() => {
    console.log('removed ' + product.productID + ' from cart!');
  });
}
```



5. **5eme Composant est product-details (fils de cart) qui contient :**

L'affichage détaillé d'un produit par ces informations complètes et leurs prix , en plus il gère l'ajout au panier .



Product-details-Html

```
<img [src]="getImageUrl(product)" [alt]="product.productTitle" />
```

```
<button class="butt" (click)="byButtonClicked(product)">Buy</button>
```

Product-details-Ts

```
@Input() product!: Product;

@Output() buy = new EventEmitter()
```

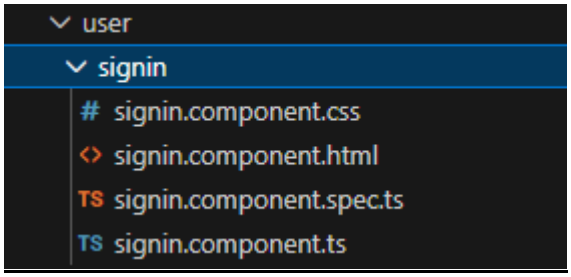
```
byButtonClicked(product: Product){
  this.buy.emit()

}
```

```
getImageUrl(product: Product){
  return '/assets/images/'+ product.productImage
}
```

6. **6eme Composant est User (sign in) qui contient :**

La gestion des utilisateurs : inscription + authentification



Sign.in-Html

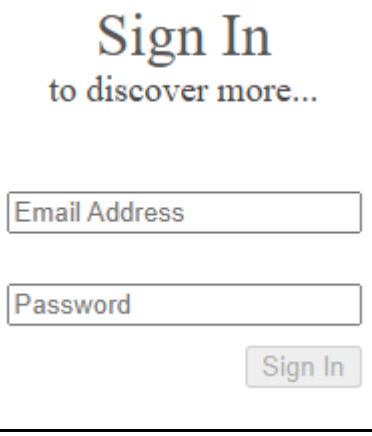
```
<input name="email" required email #email="ngModel" [(ngModel)]="credentials.email" placeholder="Email Address"
  type="text" />
<em class="error" *ngIf="email.errors?.['required'] && email.touched">Email is required</em>
<em class="error" *ngIf="email.errors?.['email'] && email.touched">Must be a valid email format</em>
```

```
<input name="password" required #password="ngModel" [(ngModel)]="credentials.password" placeholder="Password"
  type="password" />
<em class="error" *ngIf="password.invalid && password.touched">Password is required</em>
<div class="buttons">
  <button type="submit" [disabled]="signInForm.form.invalid" class="button cta">
    Sign In
  </button>
</div>
<div class="signInError" *ngIf="signInError">
  Sign In Error !!. Please try again.
</div>
```

Sign.in-Ts (En utilisent les services)

```
signIn() {
  this.signInError = false;
  this.userService.signIn(this.credentials).subscribe({
    next: () => this.router.navigate(['/catalog']),
    error: () => (this.signInError = true)
  });
}
```

```
signIn(credentials: IUserCredentials): Observable<IUser> {
  return this.http
    .post<IUser>(`${this.baseUrl}/signin`, credentials)
    .pipe(map((user: IUser) => {
      this.user.next(user);
      return user;
    }));
}
```



➢ **! Depuis l'API**


```
const users = {
  "mouadmounad@test.com": {
    userId: 1,
    firstName: "mouad",
    lastName: "mounad",
    email: "mouadmounad@test.com",
    password: "0000",
  }
};
```

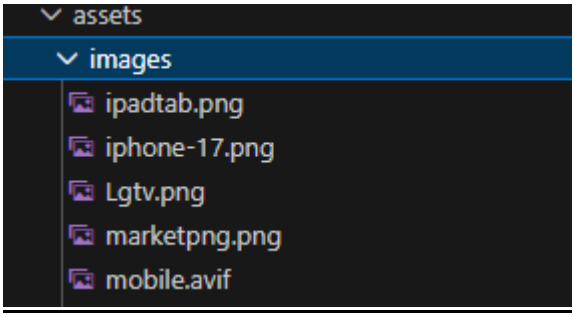
(! sign out depuis le home)

```
<button (click)="signOut()">Sign Out</button>
```

```
signOut() {
  this.user.next(null);
}
```

7. **Fichier asset :**

Ce dossier contient les ressources << images , icones , logo >> D'application .



8. **App Routes (Depuis le composant app) qui contient :**

Ce fichier est responsable de la définition des chemins et l'association entre des chemins et les composants pour la gestion de la navigation entre la page d'accueil , catalogue et cart .

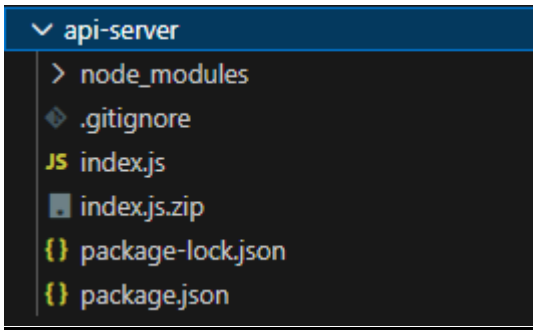
App-routes-Ts

```
export const routes: Routes = [

    {path : 'home', component : HomeComponent, title: 'My Home'},
    {path : 'catalog', component : CatalogComponent, title: 'My Catalog products'},
    {path : 'product-details', component : ProductDetailsComponent, title: 'Product details'},
    {path : 'cart', component : CartComponent, title: 'My cart'},
    {path : 'signin', component : SigninComponent, title: 'My signin page'},
    {path : '', redirectTo : '/home', pathMatch : 'full'},
];
```

9. **Fichier Api Server :**

- **Le serveur API de l'application est responsable du chargement des produits vers le frontend.**



product

```
app.get("/api/products", (req, res) => {
  let products = [
    {
      productID: "001",
      productTitle: "Samsung Tab",
      productDescription: "Description du produit",
      productImage: "samsung-tab-12.png",
      category: "tablet",
      prouctPrice: "2400 DH",
    },
```

Cart

```
app.post("/api/cart", (req, res) => {
  cart = req.body;
  res.status(201).json({ message: "Cart updated successfully" });
});

app.get("/api/cart", (req, res) => {
  res.json(cart);
});
```

Sign in

```
app.post("/api/signin", (req, res) => {
  const { email, password } = req.body;
  const user = users[email];

  if (user && user.password === password) {
    res.status(200).json({
      userId: user.userId,
      firstName: user.firstName,
      lastName: user.lastName,
      email: user.email,
    });
  } else {
    res.status(401).json({ message: "Invalid user credentials" });
  }
});
```

Api server listening on port (3000)

```
const port = 3000;

app.listen(port, () => console.log(`API Server listening on port ${port}`));
```

➢ Conclusion

Pour conclure, ce projet a permis d'appliquer les concepts de base du Framework Angular. Cela inclut la création et la gestion des composants, la communication entre les différentes parties de l'application via les services, et la mise en place du routage pour assurer une navigation fluide et organisée.