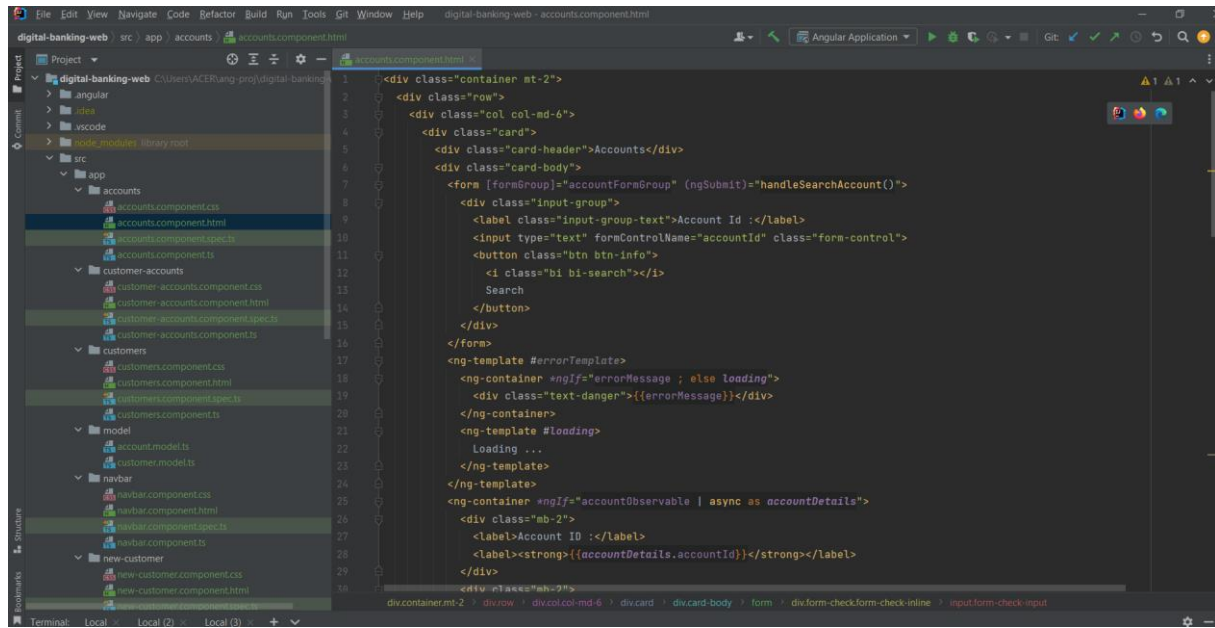
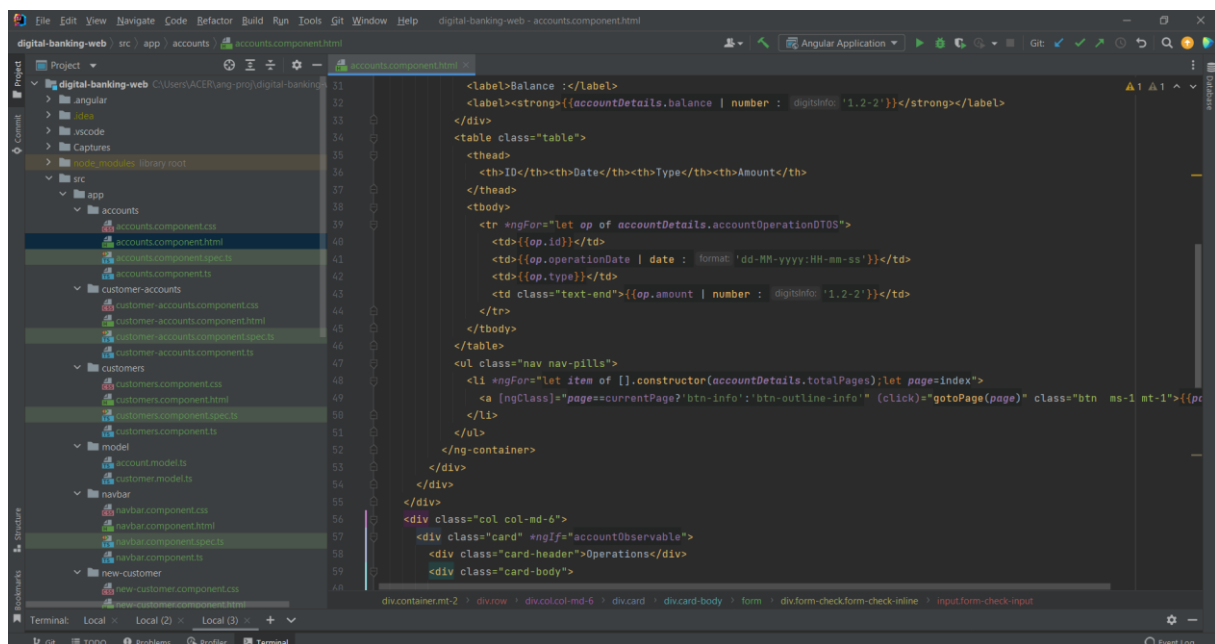


# Spring Angular Use case Digital Banking (Partie 2)

## accounts



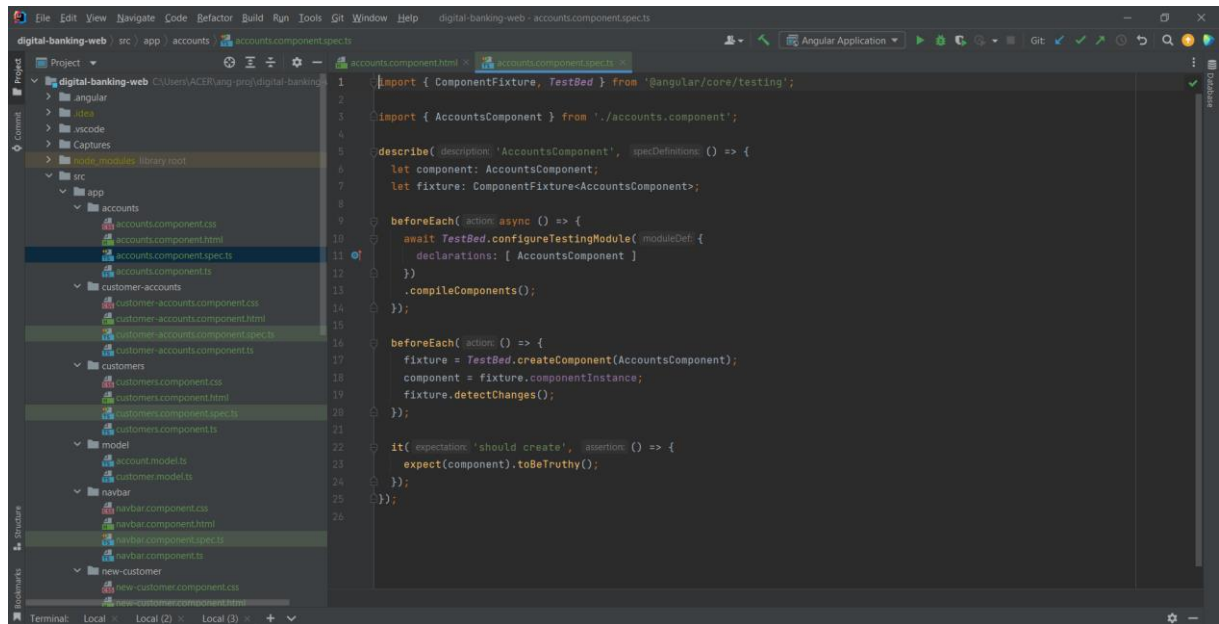
```
1 <div class="container mt-2">
2   <div class="row">
3     <div class="col col-md-6">
4       <div class="card">
5         <div class="card-header">Accounts</div>
6         <div class="card-body">
7           <form [formGroup]="accountFormGroup" (ngSubmit)="handleSearchAccount()">
8             <div class="input-group">
9               <label class="input-group-text">Account ID :</label>
10              <input type="text" formControlName="accountId" class="form-control">
11              <button class="btn btn-info">
12                <i class="bi bi-search"></i>
13                Search
14              </button>
15            </div>
16          </form>
17          <ng-template #errorTemplate>
18            <ng-container *ngIf="errorMessage ; else loading">
19              <div class="text-danger">{{errorMessage}}</div>
20            </ng-container>
21            <ng-template #loading>
22              Loading ...
23            </ng-template>
24          </ng-template>
25          <ng-container *ngIf="accountObservable | async as accountDetails">
26            <div class="mb-2">
27              <label>Account ID :</label>
28              <label><strong>{{accountDetails.accountId}}</strong></label>
29            </div>
30          </ng-container>
31        </div>
32      </div>
33    </div>
34  </div>
35</div>
```



```
31 <label>Balance :</label>
32 <label><strong>{{accountDetails.balance | number : '1.2-2'}}</strong></label>
33 </div>
34 <table class="table">
35   <thead>
36     <tr>
37       <th>ID</th>
38       <th>Date</th>
39       <th>Type</th>
40       <th>Amount</th>
41     </tr>
42   </thead>
43   <tbody>
44     <tr *ngFor="let op of accountDetails.accountOperations">
45       <td>{{op.id}}</td>
46       <td>{{op.operationDate | date : 'dd-MM-yyyy:HH-mm-ss'}}</td>
47       <td>{{op.type}}</td>
48       <td class="text-end">{{op.amount | number : '1.2-2'}}</td>
49     </tr>
50   </tbody>
51 </table>
52 <ul class="nav nav-pills">
53   <li *ngFor="let item of [].constructor(accountDetails.totalPages); let page=index">
54     <a [ngClass]="page==currentPage?'btn-info':'btn-outline-info'" (click)="gotoPage(page)" class="btn ms-1 mt-1">{{page}}</a>
55   </li>
56 </ul>
57 </ng-container>
58 </div>
59 </div>
60 <div class="col col-md-6">
61   <div class="card" *ngIf="accountObservable">
62     <div class="card-header">Operations</div>
63     <div class="card-body">
```

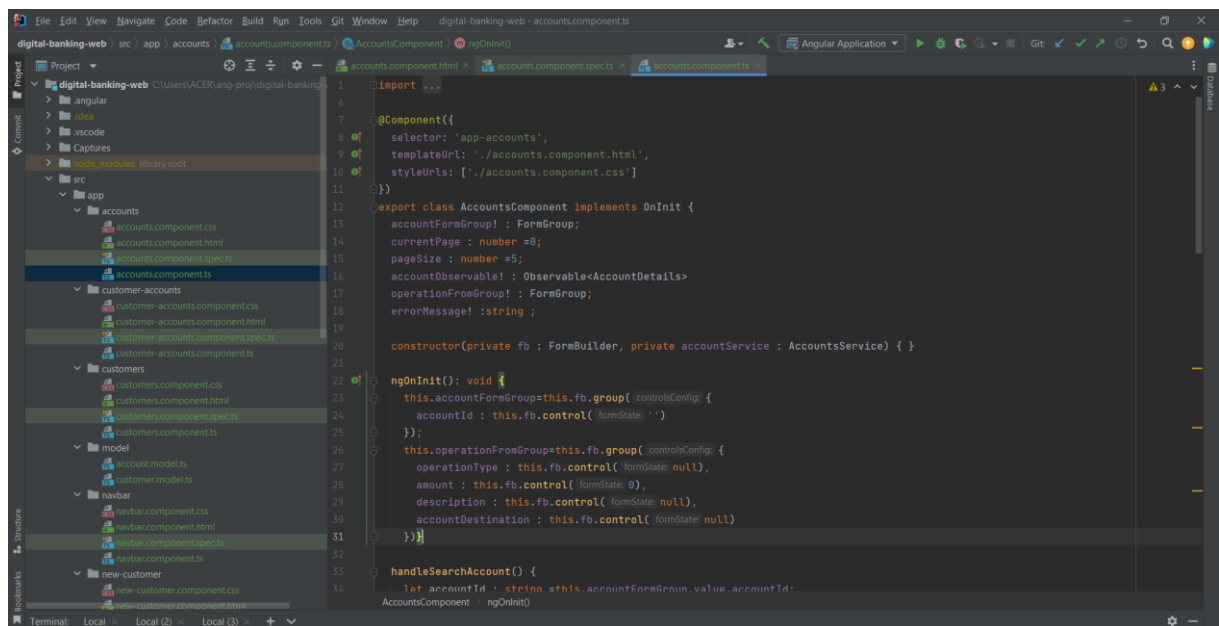
The screenshot displays an IDE with the following components:

- File Explorer (Left):** Shows the project structure for `digital-banking-web`. The `src/app/accounts` directory is expanded, listing files like `accounts.component.css`, `accounts.component.html`, `accounts.component.spec.ts`, and `accounts.component.ts`.
- Code Editor (Right):** Displays the `accounts.component.html` file. The code is an Angular template using `<form>` and `<div>` elements. It includes:
  - A form group `operationFromGroup` with a submit button `handleAccountOperation()`.
  - Three radio buttons for `operationType` with values `DEBIT`, `CREDIT`, and `TRANSFER`.
  - A text input for `accountDestination` (labeled "Account Destination").
  - A text input for `amount` (labeled "Amount").
  - A text input for `description` (labeled "Description").
  - A `<button>` labeled "Save Operation".
- Terminal (Bottom):** Shows the current terminal session with the path `Local (2)`.



This screenshot shows the VS Code editor with the file `accounts.component.spec.ts` open. The left sidebar displays the project structure of `digital-banking-web`, including folders for `src`, `node_modules`, and various component-specific directories like `accounts`, `customer-accounts`, `customers`, `model`, `nav-bar`, and `new-customer`. The main editor area contains the following TypeScript code:

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { AccountsComponent } from './accounts.component';
4
5 describe('AccountsComponent', specDefinitions() => {
6   let component: AccountsComponent;
7   let fixture: ComponentFixture<AccountsComponent>;
8
9   beforeEach(async () => {
10     await TestBed.configureTestingModule({
11       declarations: [ AccountsComponent ]
12     })
13     .compileComponents();
14   });
15
16   beforeEach(() => {
17     fixture = TestBed.createComponent(AccountsComponent);
18     component = fixture.componentInstance;
19     fixture.detectChanges();
20   });
21
22   it('expectation: \'should create\', assertion: () => {
23     expect(component).toBeTruthy();
24   });
25 });
26
```



This screenshot shows the VS Code editor with the file `accounts.component.ts` open. The left sidebar shows the same project structure as the previous image. The main editor area contains the following TypeScript code:

```
1 import ...
2
3 @Component({
4   selector: 'app-accounts',
5   templateUrl: './accounts.component.html',
6   styleUrls: ['./accounts.component.css']
7 })
8
9 export class AccountsComponent implements OnInit {
10   accountFormGroup! : FormGroup;
11   currentPage : number = 0;
12   pageSize : number = 5;
13   accountObservable! : Observable<AccountDetails>
14   operationFormGroup! : FormGroup;
15   errorMessage! : string ;
16
17   constructor(private fb : FormBuilder, private accountService : AccountsService) { }
18
19   ngOnInit(): void {
20     this.accountFormGroup=this.fb.group( controlsConfig: {
21       accountId : this.fb.control( formState: '' )
22     });
23     this.operationFormGroup=this.fb.group( controlsConfig: {
24       operationType : this.fb.control( formState: null),
25       amount : this.fb.control( formState: 0),
26       description : this.fb.control( formState: null),
27       accountDestination : this.fb.control( formState: null)
28     });
29   }
30
31   handleSearchAccount() {
32     let accountId = <string>this.accountFormGroup.value.accountId;
33     AccountsComponent.ngOnInit()
34   }
35 }

```



The screenshot shows the Visual Studio Code editor with the file `customer-accounts.component.html` open. The editor displays the following HTML code:

```
1 <div class="container">
2   <div>{{customerId}}</div>
3   <div>{{customer | json}}</div>
4 </div>
```

The left sidebar shows the project structure for `digital-banking-web`, with the `customer-accounts` folder selected. The bottom status bar indicates the file is part of an `Angular Application`.

The screenshot shows the Visual Studio Code editor with the file `customer-accounts.component.spec.ts` open. The editor displays the following TypeScript code:

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { CustomerAccountsComponent } from './customer-accounts.component';
4
5 describe('CustomerAccountsComponent', specDefinitions: () => {
6   let component: CustomerAccountsComponent;
7   let fixture: ComponentFixture<CustomerAccountsComponent>;
8
9   beforeEach(async () => {
10     await TestBed.configureTestingModule({
11       declarations: [ CustomerAccountsComponent ]
12     })
13     .compileComponents();
14   });
15
16   beforeEach(() => {
17     fixture = TestBed.createComponent(CustomerAccountsComponent);
18     component = fixture.componentInstance;
19     fixture.detectChanges();
20   });
21
22   it('should create', () => {
23     expect(component).toBeTruthy();
24   });
25 });
```

The left sidebar shows the project structure for `digital-banking-web`, with the `customer-accounts` folder selected. The bottom status bar indicates the file is part of an `Angular Application`.

The screenshot shows the VS Code editor with the file `customer-accounts.component.ts` open. The code defines an Angular component for managing customer accounts. The component has a selector `app-customer-accounts`, a template URL `./customer-accounts.component.html`, and a style URL `./customer-accounts.component.css`. It implements the `OnInit` interface, with a `customerId` property of type `string`. The constructor takes an `ActivatedRoute` and a `Router` as arguments, and sets `this.customer` to the state from the current navigation. The `ngOnInit` method sets `this.customerId` to the `id` parameter from the route snapshot.

```
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-customer-accounts',
5   templateUrl: './customer-accounts.component.html',
6   styleUrls: ['./customer-accounts.component.css']
7 })
8
9 export class CustomerAccountsComponent implements OnInit {
10   customerId: string;
11   customer!: Customer;
12   constructor(private route: ActivatedRoute, private router: Router) {
13     this.customer = this.router.getCurrentNavigation()?.extras.state as Customer;
14   }
15
16   ngOnInit(): void {
17     this.customerId = this.route.snapshot.params['id'];
18   }
19 }
20
21
22
23
```

## customers

The screenshot shows the VS Code editor with the file `customers.component.html` open. The template displays a search form and a table of customers. The search form includes a text input for the keyword and a search button. The table lists customers with columns for ID, Name, and Email, and includes a delete button for each row. The code uses Angular's `*ngIf` and `*ngFor` directives for conditional rendering and iteration.

```
1 <div class="container mt-2">
2   <ng-container *ngIf="customers | async as listCustomers; else failureOrLoading">
3     <div class="card">
4       <div class="card-header">Customers</div>
5       <div class="card-body">
6         <div *ngIf="searchFormGroup">
7           <form [formGroup]="searchFormGroup" (ngSubmit)="handleSearchCustomers()">
8             <div class="input-group">
9               <label class="input-group-text">Keyword </label>
10              <input type="text" formControlName="keyword" class="form-control">
11              <button class="btn btn-info">
12                <i class="bi bi-search"></i>
13              </button>
14            </div>
15          </form>
16        </div>
17        <table class="table">
18          <thead>
19            <tr>
20              <th>ID</th>
21              <th>Name</th>
22              <th>Email</th>
23            </tr>
24          </thead>
25          <tbody>
26            <tr *ngFor="let c of customers | async">
27              <td>{{c.id}}</td>
28              <td>{{c.name}}</td>
29              <td>{{c.email}}</td>
30              <td>
31                <button (click)="handleDeleteCustomer(c)" class="btn btn-danger">
32                  <i class="bi bi-trash"></i>
33                </button>
34              </td>
35            </tr>
36          </tbody>
37        </table>
38      </div>
39    </div>
40  </ng-container>
41 </div>
```

The screenshot shows the Visual Studio Code editor with the file `customers.component.html` open. The left sidebar displays the project structure for `digital-banking-web`, with the `customers` folder selected. The main editor area contains the following HTML code:

```
27 <td>{{c.email}}</td>
28 <td>
29 <button (click)="handleDeleteCustomer(c)" class="btn btn-danger">
30 <i class="bi bi-trash"></i>
31 </button>
32 </td>
33 <td>
34 <button (click)="handleCustomerAccounts(c)" class="btn btn-success">
35 Accounts
36 </button>
37 </td>
38 </tr>
39 </tbody>
40 </table>
41 </div>
42 </ng-container>
43 <ng-template #failureOrLoading>
44 <ng-container #ngIf="errorMessage; else loading">
45 <div class="text-danger">
46 {{errorMessage}}
47 </div>
48 </ng-container>
49 </ng-template #loading>
50 <ng-template #loading>
51 Loading .....
52 </ng-template>
53 </ng-template>
54 </div>
```

The bottom status bar indicates the current context: `div.container.mt-2 > ng-container > div.card > div.card-body > table.table > tbody > tr`. A notification at the bottom right states: "Externally added files can be added to Git" with options "View Files", "Always Add", and "Don't Ask Again".

The screenshot shows the Visual Studio Code editor with the file `customers.component.spec.ts` open. The left sidebar displays the project structure for `digital-banking-web`, with the `customers` folder selected. The main editor area contains the following TypeScript code:

```
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { CustomersComponent } from './customers.component';
4
5 describe('CustomersComponent', specDefinitions() => {
6   let component: CustomersComponent;
7   let fixture: ComponentFixture<CustomersComponent>;
8
9   beforeEach(async () => {
10     await TestBed.configureTestingModule({
11       declarations: [ CustomersComponent ]
12     }).compileComponents();
13   });
14
15   beforeEach(() => {
16     fixture = TestBed.createComponent(CustomersComponent);
17     component = fixture.componentInstance;
18     fixture.detectChanges();
19   });
20
21   it('expectation: 'should create'', assertion() => {
22     expect(component).toBeTruthy();
23   });
24 });
25
26
```

The bottom status bar indicates the current context: `div.container.mt-2 > ng-container > div.card > div.card-body > table.table > tbody > tr`. A notification at the bottom right states: "Externally added files can be added to Git" with options "View Files", "Always Add", and "Don't Ask Again".

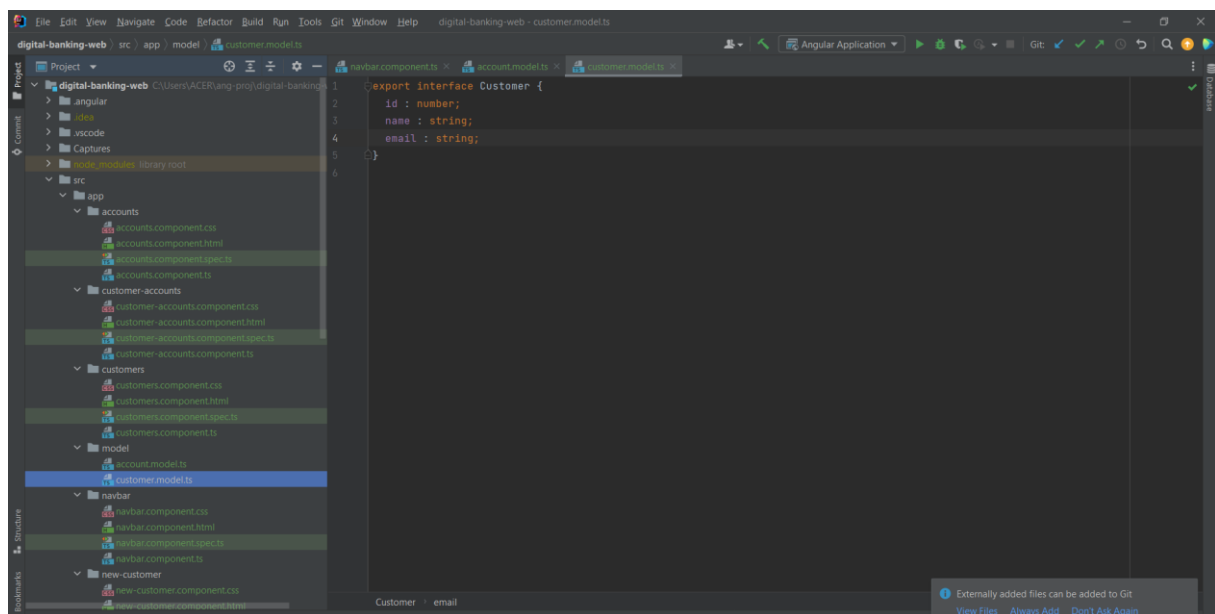
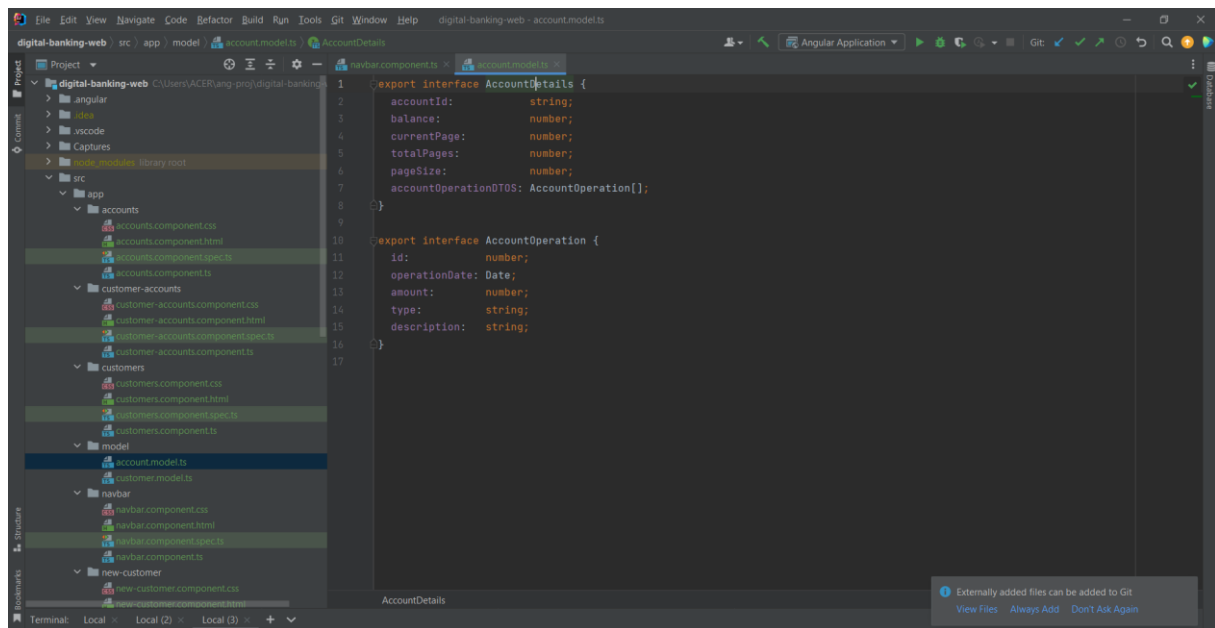


```
1 import { Component, OnInit } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { CustomerService } from '../services/customer.service';
4 import { catchError, map, Observable, throwError } from 'rxjs';
5 import { Customer } from '../model/customer.model';
6 import { FormBuilder, FormGroup } from '@angular/forms';
7 import { Router } from '@angular/router';
8
9 @Component({
10   selector: 'app-customers',
11   templateUrl: './customers.component.html',
12   styleUrls: ['./customers.component.css']
13 })
14 export class CustomersComponent implements OnInit {
15   customers!: Observable<Array<Customer>>;
16   errorMessage!: string;
17   searchFormGroup!: FormGroup | undefined;
18   constructor(private customerService : CustomerService, private fb : FormBuilder, private router : Router) { }
19
20   ngOnInit(): void {
21     this.searchFormGroup=this.fb.group({ controlConfig: {
22       keyword : this.fb.control( formState: '')
23     }});
24     this.handleSearchCustomers();
25   }
26   handleSearchCustomers() {
27     let kw=this.searchFormGroup?.value.keyword;
28     this.customers=this.customerService.searchCustomers(kw).pipe(
29       catchError( selector: err => {
30         this.errorMessage=err.message;
```

```
31     this.errorMessage=err.message;
32     return throwError(err);
33   });
34 }
35
36 handleDeleteCustomer(c: Customer) {
37   let conf = confirm("Are you sure?");
38   if(!conf) return;
39   this.customerService.deleteCustomer(c.id).subscribe( observer: {
40     next : (resp :Object ) => {
41       this.customers=this.customers.pipe(
42         map( project: data=>{
43           let index=data.indexOf(c);
44           data.slice(index,1)
45           return data;
46         })
47       );
48     },
49     error : err => {
50       console.log(err);
51     }
52   })
53 }
54
55 handleCustomerAccounts(customer: Customer) {
56   this.router.navigateByUrl( url: "/customer-accounts/"+customer.id, extra {state :customer});
57 }
58
59 }
```

model

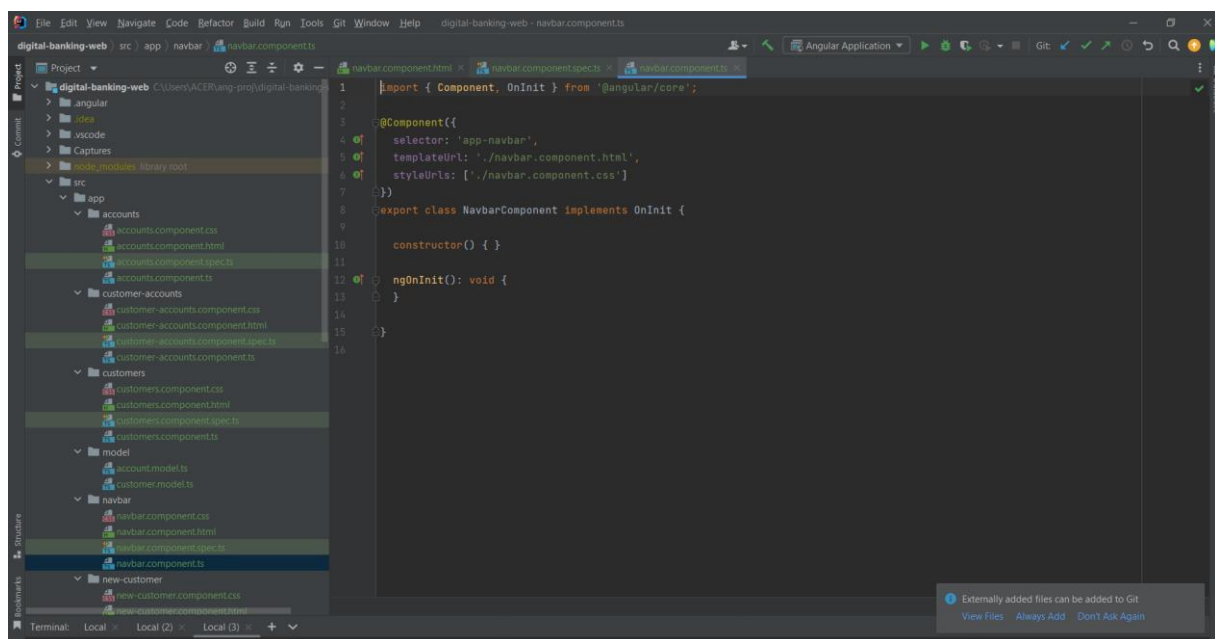
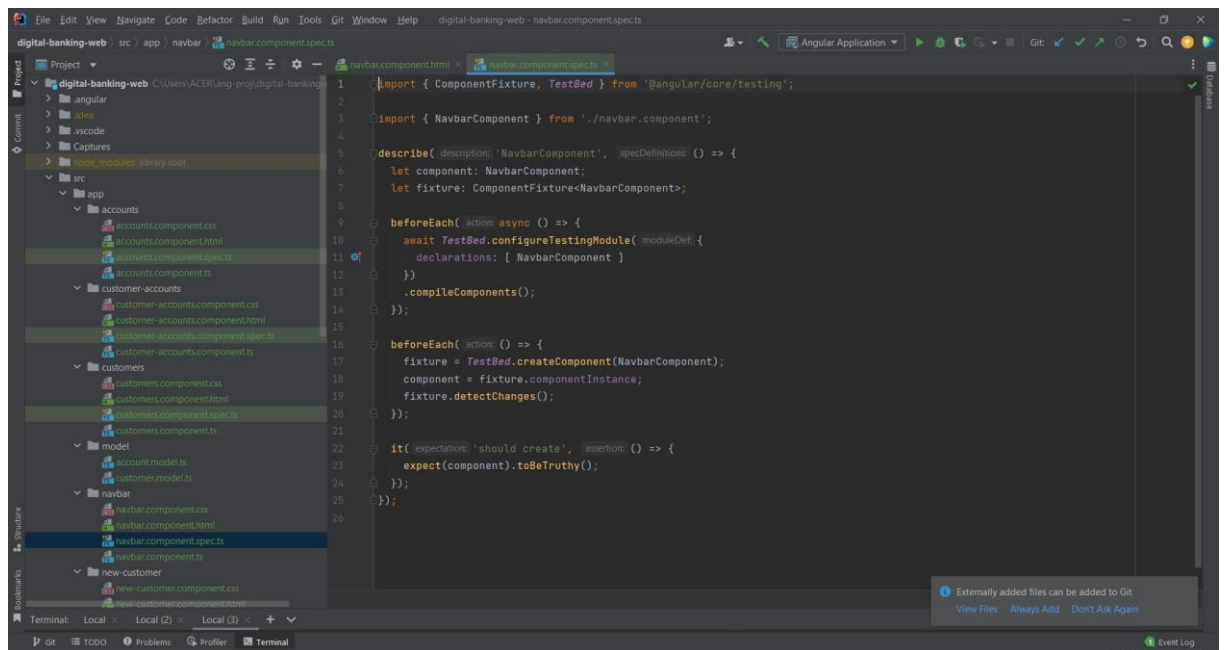




navbar

```
1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2   <div class="container-fluid">
3     <a class="navbar-brand" href="#">Navbar</a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="
5       <span class="navbar-toggler-icon"></span>
6     </button>
7     <div class="collapse navbar-collapse" id="navbarSupportedContent">
8       <ul class="navbar-nav me-auto mb-2 mb-lg-0">
9         <li class="nav-item">
10          <a class="nav-link active" aria-current="page" href="#">Home</a>
11        </li>
12        <li class="nav-item">
13          <a class="nav-link" routerLink="/accounts">Accounts</a>
14        </li>
15        <li class="nav-item dropdown">
16          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="f
17            Customers
18          </a>
19          <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
20            <li><a class="dropdown-item" routerLink="/customers">Search customers</a></li>
21            <li><a class="dropdown-item" routerLink="/new-customer">New customer</a></li>
22          </ul>
23        </li>
24        <li class="nav-item">
25          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
26        </li>
27      </ul>
28      <form class="d-flex">
29        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
30        <button class="btn btn-outline-success" type="submit">Search</button>
31      </form>
32    </div>
33  </div>
34</nav>
```

```
12 <li class="nav-item">
13   <a class="nav-link" routerLink="/accounts">Accounts</a>
14 </li>
15 <li class="nav-item dropdown">
16   <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="f
17     Customers
18   </a>
19   <ul class="dropdown-menu" aria-labelledby="navbarDropdown">
20     <li><a class="dropdown-item" routerLink="/customers">Search customers</a></li>
21     <li><a class="dropdown-item" routerLink="/new-customer">New customer</a></li>
22   </ul>
23 </li>
24 <li class="nav-item">
25   <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
26 </li>
27 </ul>
28 <form class="d-flex">
29   <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
30   <button class="btn btn-outline-success" type="submit">Search</button>
31 </form>
32 </div>
33 </div>
34</nav>
```



## new-customer

```
digital-banking-web - new-customer.component.html
1 <div class="container">
2   <div class="card col-md-6 offset-3 mt-2">
3     <div class="card-header">New Customer</div>
4     <div class="card-body">
5       <form [formGroup]="newCustomerFormGroup" (ngSubmit)="handleSaveCustomer()">
6         <div class="mb-3">
7           <label class="form-label">Name:</label>
8           <input type="text" formControlName="name" class="form-control">
9           <span class="text-danger">
10             *ngIf="newCustomerFormGroup.controls['name'].touched
11               && newCustomerFormGroup.controls['name'].errors
12               && newCustomerFormGroup.controls['name'].errors['required']">
13             Name is Required
14           </span>
15         </div>
16         <div class="mb-3">
17           <label class="form-label">Email:</label>
18           <input type="text" formControlName="email" class="form-control">
19           <span class="text-danger">
20             *ngIf="newCustomerFormGroup.controls['email'].touched
21               && newCustomerFormGroup.controls['email'].errors
22               && newCustomerFormGroup.controls['email'].errors['email']">
23             Email is not valid
24           </span>
25         </div>
26         <button [disabled]="!newCustomerFormGroup.valid" class="btn btn-info">Save</button>
27       </form>
28     </div>
29   </div>
30 </div>
```

```
digital-banking-web - new-customer.component.spec.ts
1 import { ComponentFixture, TestBed } from '@angular/core/testing';
2
3 import { NewCustomerComponent } from './new-customer.component';
4
5 describe('NewCustomerComponent', () => {
6   let component: NewCustomerComponent;
7   let fixture: ComponentFixture<NewCustomerComponent>;
8
9   beforeEach(async () => {
10     await TestBed.configureTestingModule({
11       declarations: [ NewCustomerComponent ]
12     })
13     .compileComponents();
14   });
15
16   beforeEach(() => {
17     fixture = TestBed.createComponent(NewCustomerComponent);
18     component = fixture.componentInstance;
19     fixture.detectChanges();
20   });
21
22   it('should create', () => {
23     expect(component).toBeTruthy();
24   });
25 });
```

The screenshot shows the VS Code editor with the `new-customer.component.ts` file open. The file implements the `NewCustomerComponent` class, which is a form for creating a new customer. The code includes imports for `Component`, `FormBuilder`, `FormGroup`, `Router`, `CustomerService`, and `Validators`. The component has a selector `app-new-customer`, a template URL, and a style URL. The `ngOnInit` method initializes the `newCustomerFormGroup` with form state and validators for required, minimum length, and email. The `handleSaveCustomer` method calls `customerService.saveCustomer` and displays a success alert, then navigates to the customers page. Error handling is implemented using `Observable` and `console.log`.

```
import { Component, OnInit, FormBuilder, FormGroup, Router, CustomerService, Validators } from '@angular/core';

@Component({
  selector: 'app-new-customer',
  templateUrl: './new-customer.component.html',
  styleUrls: ['./new-customer.component.css']
})
export class NewCustomerComponent implements OnInit {
  newCustomerFormGroup!: FormGroup;
  constructor(private fb : FormBuilder, private customerService:CustomerService, private router:Router) {}

  ngOnInit(): void {
    this.newCustomerFormGroup=this.fb.group( controlsConfig: {
      name : this.fb.control( formState: null, validatorOptions: [Validators.required, Validators.minLength( minLength: 4)] ),
      email : this.fb.control( formState: null, validatorOptions: [Validators.required, Validators.email] )
    });
  }

  handleSaveCustomer() {
    let customer:Customer=this.newCustomerFormGroup.value;
    this.customerService.saveCustomer(customer).subscribe( observer: {
      next : data=>{
        alert("Customer has been successfully saved!");
        //this.newCustomerFormGroup.reset();
        this.router.navigateByUrl( url: "/customers");
      },
      error : err => {
        console.log(err);
      }
    });
  }
}
```

## services

The screenshot shows the VS Code editor with the `accounts.service.ts` file open. The file implements the `AccountsService` class, which provides methods for interacting with the accounts API. The service is decorated with `@Injectable` and has a `providedIn: 'root'` configuration. The `constructor` takes an `HttpClient` instance. The methods include `getAccount`, `debit`, `credit`, and `transfer`, all of which use `HttpClient` to make HTTP requests to the backend API.

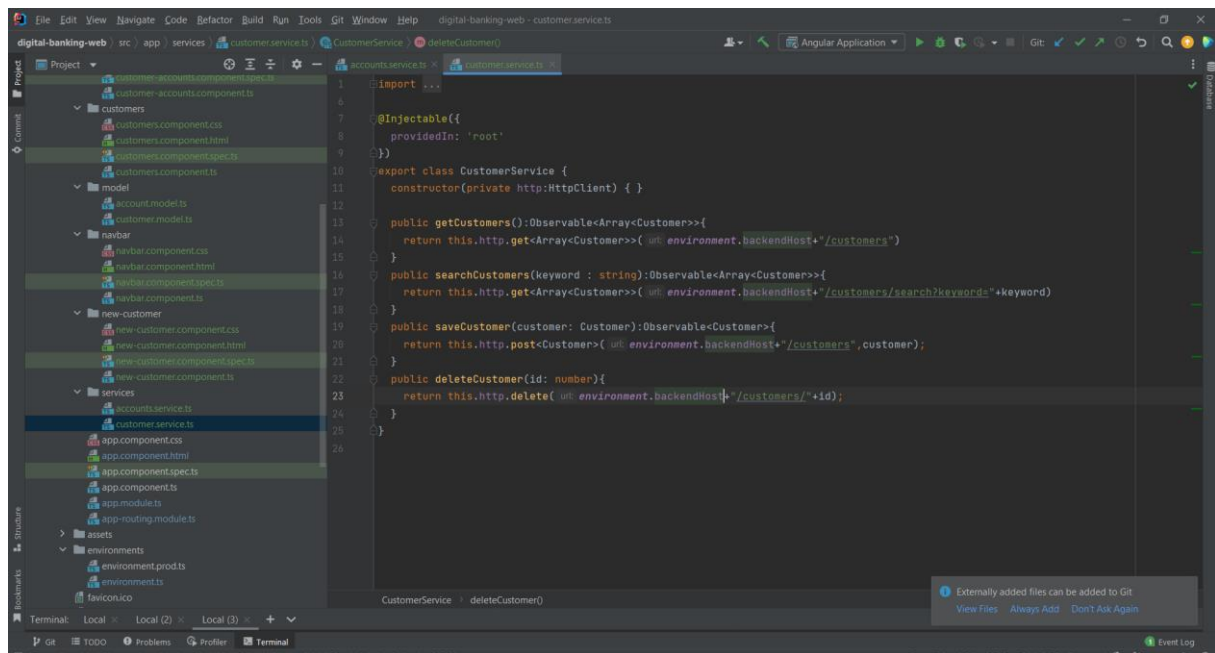
```
@Injectable({
  providedIn: 'root'
})
export class AccountsService {
  constructor(private http : HttpClient) {}

  public getAccount(accountId : string, page : number, size : number):Observable<AccountDetails>{
    return this.http.get<AccountDetails>(`${environment.backendHost}/accounts/${accountId}/pageOperations?page=${page}&size=${size}`);
  }

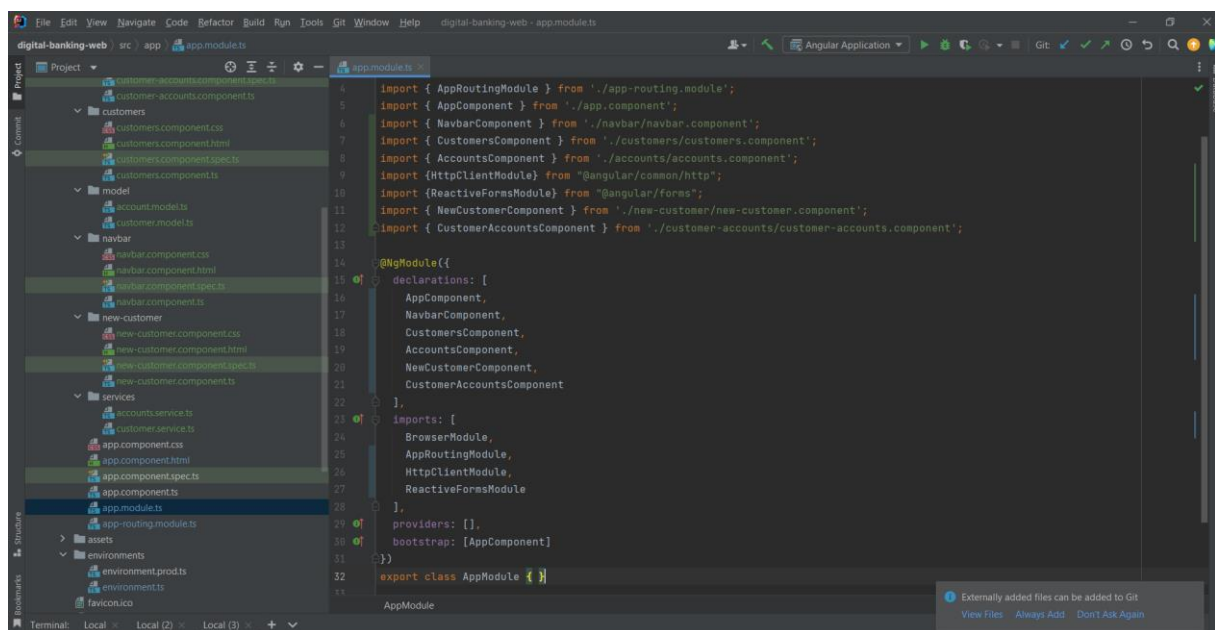
  public debit(accountId : string, amount : number, description:string){
    let data={accountId : accountId, amount : amount, description : description};
    return this.http.post( url: environment.backendHost+"/accounts/debit",data);
  }

  public credit(accountId : string, amount : number, description:string){
    let data={accountId : accountId, amount : amount, description : description};
    return this.http.post( url: environment.backendHost+"/accounts/credit",data);
  }

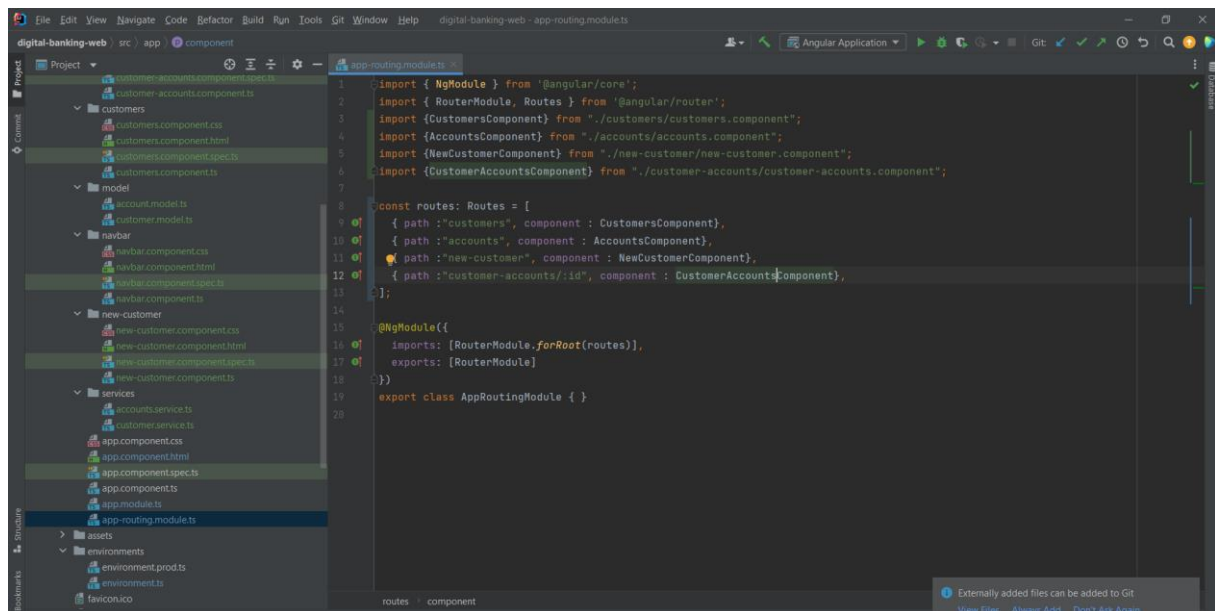
  public transfer(accountSource: string,accountDestination: string, amount : number, description:string){
    let data={accountSource, accountDestination, amount, description };
    return this.http.post( url: environment.backendHost+"/accounts/transfer",data);
  }
}
```



## app

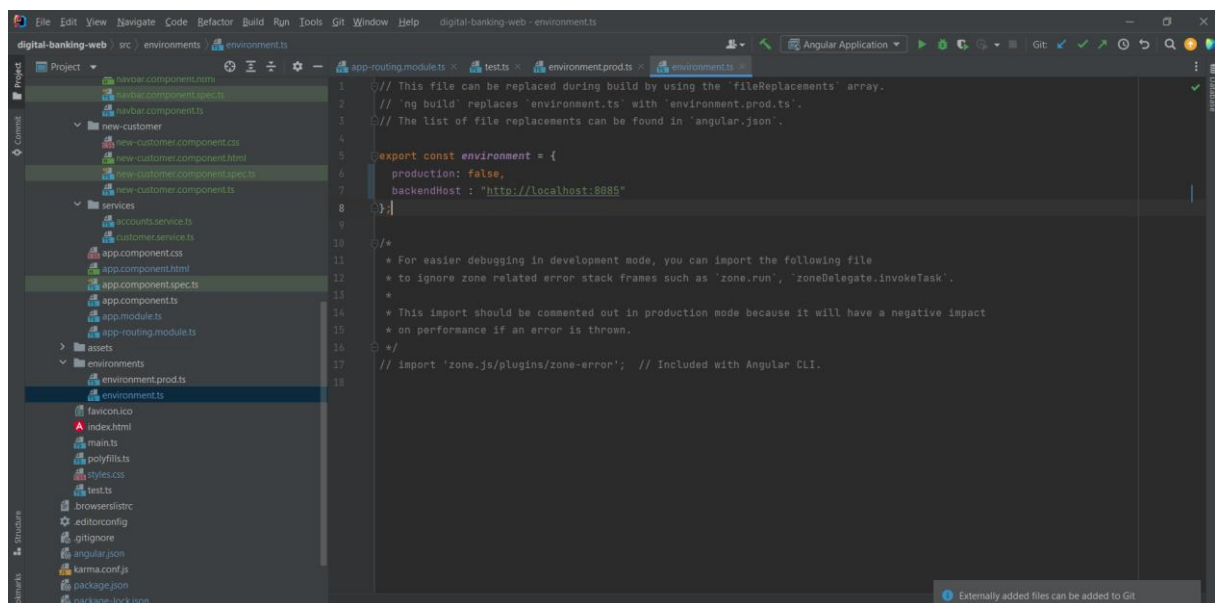






The screenshot shows the Visual Studio Code editor with the 'digital-banking-web' project open. The file explorer on the left displays the project structure, including folders for 'customers', 'model', 'new-customer', 'services', 'assets', and 'environments'. The main editor window displays the 'app-routing.module.ts' file, which contains the following code:

```
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { CustomersComponent } from './customers/customers.component';
4 import { AccountsComponent } from './accounts/accounts.component';
5 import { NewCustomerComponent } from './new-customer/new-customer.component';
6 import { CustomerAccountsComponent } from './customer-accounts/customer-accounts.component';
7
8 const routes: Routes = [
9   { path: 'customers', component: CustomersComponent },
10  { path: 'accounts', component: AccountsComponent },
11  { path: 'new-customer', component: NewCustomerComponent },
12  { path: 'customer-accounts/:id', component: CustomerAccountsComponent },
13 ];
14
15 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule]
18 })
19 export class AppRoutingModule { }
20
```



The screenshot shows the Visual Studio Code editor with the 'digital-banking-web' project open. The file explorer on the left displays the project structure, including folders for 'new-customer', 'services', 'assets', and 'environments'. The main editor window displays the 'environments.ts' file, which contains the following code:

```
1 /** This file can be replaced during build by using the 'fileReplacements' array.
2  *  * 'ng build' replaces 'environments.ts' with 'environment.prod.ts'.
3  *  * The list of file replacements can be found in 'angular.json'.
4  */
5
6 export const environment = {
7   production: false,
8   backendHost: 'http://localhost:8085'
9 };
10
11 /*
12  * For easier debugging in development mode, you can import the following file
13  * to ignore zone related error stack frames such as 'zone.run', 'zoneDelegate.invokeTask'.
14  * This import should be commented out in production mode because it will have a negative impact
15  * on performance if an error is thrown.
16  */
17 // import 'zone.js/plugins/zone-error'; // Included with Angular CLI.
18
```

## Web



Part 4 Spring Angular Use case | DigitalBankingWeb | digital-banking-spring-backend | digital-banking-angular-front/ | localhost / 127.0.0.1 / e-bank / |

localhost:4200/customers

Navbar Home Accounts Customers Disabled

Aicha Search

Customers

Keyword:

ID	Name	Email		
4	Hassan	Hassan@gmail.com	<input type="button" value="Delete"/>	<input type="button" value="Accounts"/>
5	Yassine	Yassine@gmail.com	<input type="button" value="Delete"/>	<input type="button" value="Accounts"/>
6	Aicha	Aicha@gmail.com	<input type="button" value="Delete"/>	<input type="button" value="Accounts"/>

Part 4 Spring Angular Use case | DigitalBankingWeb | digital-banking-spring-backend | digital-banking-angular-front/ | localhost / 127.0.0.1 / e-bank / |

localhost:4200/customers

Navbar Home Accounts Customers Disabled

Aicha Search

Customers

Keyword: Hassan

ID	Name	Email		
4	Hassan	Hassan@gmail.com	<input type="button" value="Delete"/>	<input type="button" value="Accounts"/>

Part 4 Spring Angular Use case | DigitalBankingWeb | digital-banking-spring-backend | digital-banking-angular-front/ | localhost / 127.0.0.1 / e-bank / |

localhost:4200/customer-accounts/4

Navbar Home Accounts Customers Disabled

Aicha Search

4

{ "id": 4, "name": "Hassan", "email": "Hassan@gmail.com" }

