# Final Project ”GoHealthy”

An e-commerce website.

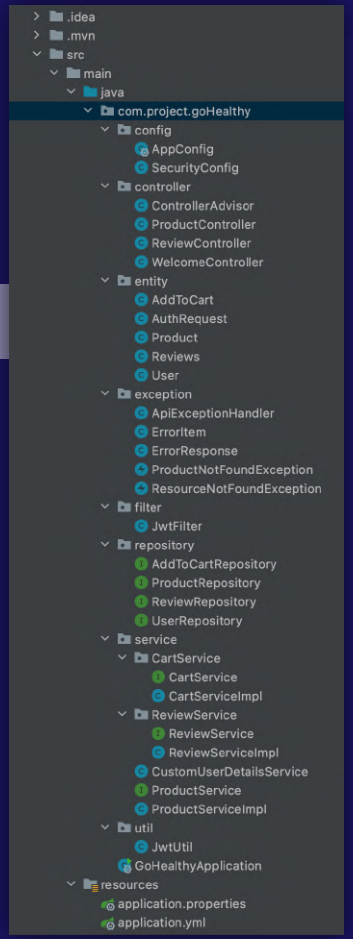
# Overview

The project consists in a full-stack e-commerce site based on food ordering.

The project includes CRUD (Create, Read, Update, Delete) operations for the products and reviews, obviously the update and the delete methods are going to be accesible only by the administrators of the site. Same thing with the add method for products.

Speaking of administrators and roles, in the project it was implemented the JWT (JSON Web Token) authentication, which provide a better security for the site.
Go Healthy is a restaurant and a fresh food provider. Besides the dish, you can order the ingredients as well, and, if you don't want to order anything you can still look at the recipe attached to every product.

# The overall structure



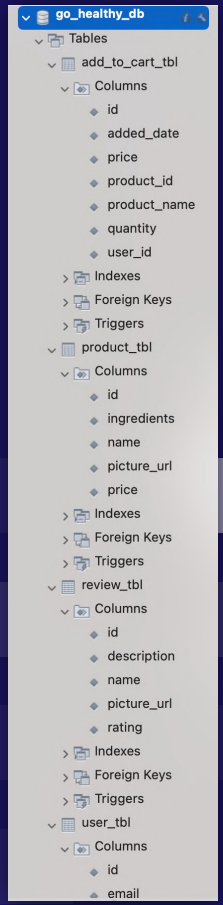

Here is the structure of the Java packages and classes:

- the controller is responsible for processing the REST API methods.

- the entity class define the object, and create the table in MySql as well.

- the service come as an interface and an implementation class, and it is used to write logic separated from controller.

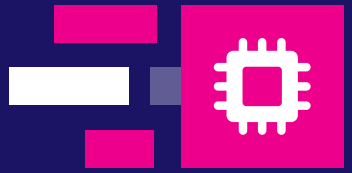
Here is the structure of the MySql database tables.



# Technologies used

## Angular

The front-end framework used for working with HTML, CSS and TypeScript.

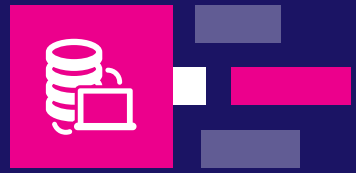


## Java

For the back-end logic.

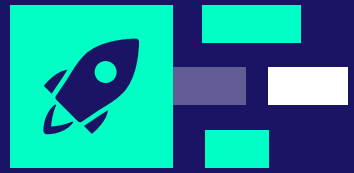
## MySql Database

The database where all the data (like users data or products data) generated by the back-end is stored.



## Spring Boot

An open-source micro framework, that is used to develop Java applications easely.



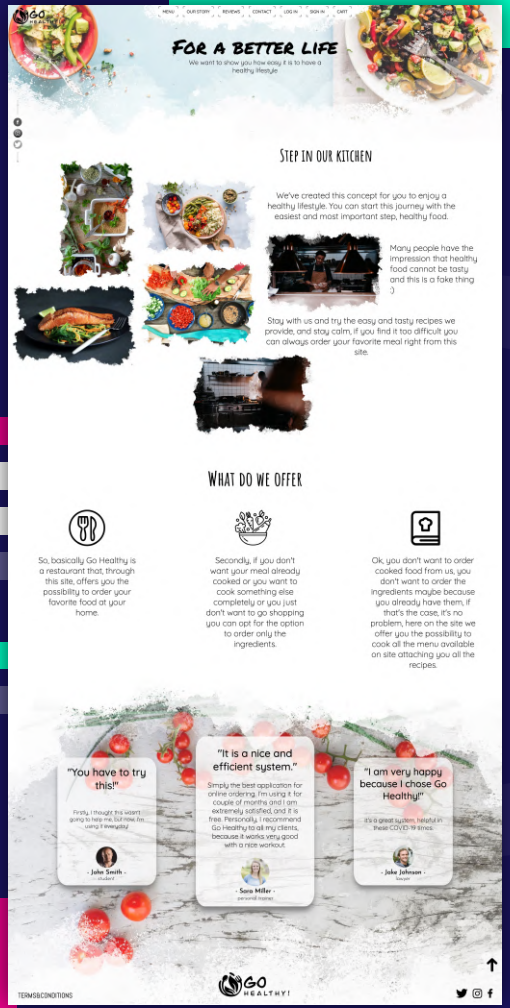
The website covers 8 main pages, Home, Menu, Our Story, Reviews, Contact, Log In, Sign In and Cart.

# 01

# Home

# The Home Component - Front-end


This is the first thing that the user sees when he opens the site. This is the first interaction with the app.




# The Home Component - Front-end

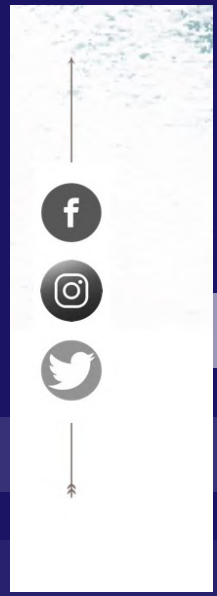
Here is the full preview of that page.
The Home page is composed of a Header, with the Navbar, Main content with important information, and Footer with reviews and social media buttons.

# In detail

```css
/* Menu */

.menu_a {
    position: relative;
    display: inline-block;
    padding: 8px 12px;
    border: 1px solid #111;
    text-transform: uppercase;
    font-family: 'Quicksand', sans-serif;
    color: #111;
    text-decoration: none;
    font-weight: 400;
    font-size: 15px;
}

.menu_a:before {
    content: '';
    position: absolute;
    top: 6px;
    left: -2px;
    width: calc(100% + 4px);
    height: calc(100% - 12px);
    background: #fff;
    transition: 0.5s ease-in-out;
    transform: scaleY(1);
    transition-delay: 0.5s;
}

.menu_a:hover::before {
    transform: scaleY(0);
}

.menu_a:after {
    content: '';
    position: absolute;
    left: 6px;
    top: -2px;
    height: calc(100% + 4px);
    width: calc(100% - 12px);
    background: #fff;
    transition: 0.5s ease-in-out;
    transform: scaleX(1);
}

.menu_a:hover::after {
    transform: scaleX(0);
}

.menu_a_span {
    position: relative;
    z-index: 3;
}
```

```html
<body>
  <a name="top"></a>
  <div class="header">
    <img class="start_photo" src="assets/images/start_photo3_6.jpg" alt="" />
    <div class="logo_container">
      <a routerLink=""
        ><img class="logo" src="assets/images/logo.png" alt=""
      /></a>
    </div>
    <div class="navbar_photo">
      <img src="assets/images/navbar_photo5.png" alt="" />
    </div>
    <div class="navbar_buttons">
      <div class="menu">
        <a class="menu_a" routerLink="/menu">
          <span class="menu_a_span">Menu</span></a
        >
      </div>

      <div class="our_story">
        <a class="our_story_a" routerLink="/our-story">
          <span class="our_story_a_span">Our story</span></a
        >
      </div>

      <div class="reviews">
        <a class="reviews_a" routerLink="/reviews">
          <span class="reviews_a_span">Reviews</span>
        </a>
      </div>

      <div class="contact">
        <a class="contact_a" routerLink="/contact">
          <span class="contact_a_span">Contact</span>
        </a>
      </div>

      <div class="log_in">
        <a class="log_in_a" routerLink="/log-in">
          <span class="log_in_a_span">Log In</span>
        </a>
      </div>
```

Here is the animation of the social media buttons.



Here we see a video with the navigation bar buttons, along with a snippet of the HTML and CSS code.

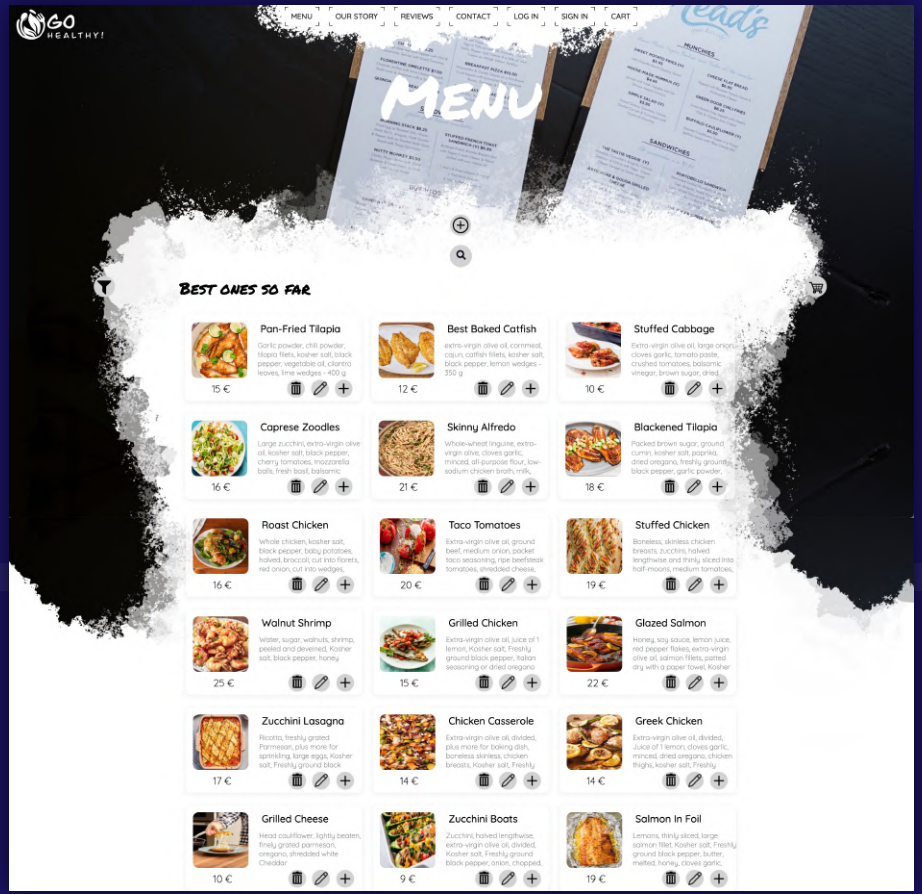
# 02

# Menu

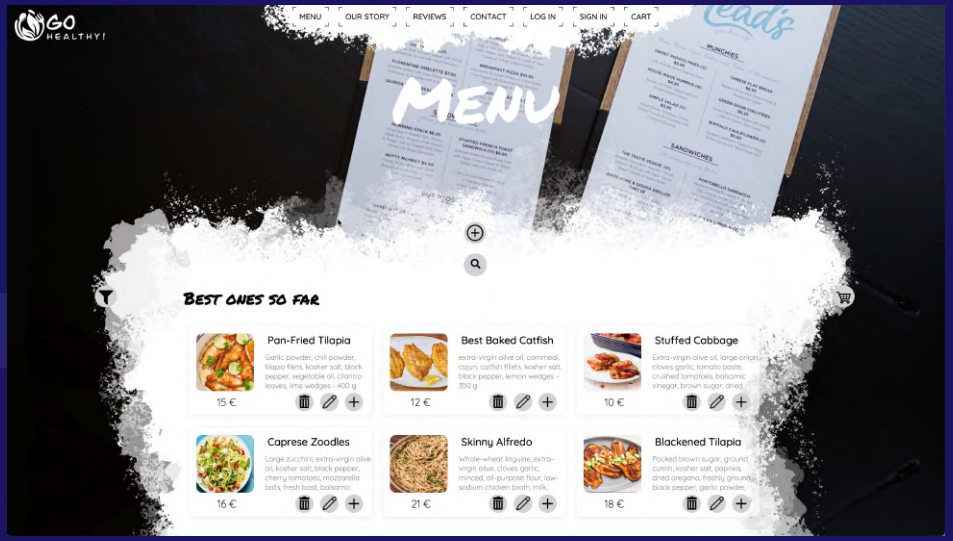
# The Menu Component - Front-end

This is the full preview of the menu page. This page consists in a button for adding a new product, a search bar, and a component that stores the products loaded from the database.

| id | ingredients | name | picture_url | price |
|---|---|---|---|---|
| 1 | Garlic powder, chili powder, tilapia filets, kosher… | Pan-Fried Tilapia | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 15 |
| 2 | extra-virgin olive oil, cornmeal, cajun, catfish fill… | Best Baked Catfish | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 12 |
| 3 | Extra-virgin olive oil, large onion, cloves garlic, t… | Stuffed Cabbage | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 10 |
| 4 | Large zucchini, extra-virgin olive oil, kosher salt,… | Caprese Zoodles | https://hips.hearstapps.com/del.h-cdn.co/assets… | 16 |
| 12 | Whole-wheat linguine, extra-virgin olive, cloves… | Skinny Alfredo | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 21 |
| 13 | Packed brown sugar, ground cumin, kosher salt,… | Blackened Tilapia | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 18 |
| 14 | Whole chicken, kosher salt, black pepper, baby… | Roast Chicken | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 16 |
| 15 | Extra-virgin olive oil, ground beef, medium onio… | Taco Tomatoes | https://hips.hearstapps.com/del.h-cdn.co/assets… | 20 |
| 16 | Boneless, skinless chicken breasts, zucchini, ha… | Stuffed Chicken | https://hips.hearstapps.com/vidthumb/images/d… | 19 |
| 17 | Water, sugar, walnuts, shrimp, peeled and devei… | Walnut Shrimp | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 25 |
| 18 | Extra-virgin olive oil, juice of 1 lemon, Kosher sa… | Grilled Chicken | https://hips.hearstapps.com/del.h-cdn.co/assets… | 15 |
| 19 | Honey, soy sauce, lemon juice, red pepper flake… | Glazed Salmon | https://hips.hearstapps.com/del.h-cdn.co/assets… | 22 |
| 20 | Ricotta, freshly grated Parmesan, plus more for… | Zucchini Lasagna | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 17 |
| 21 | Extra-virgin olive oil, divided, plus more for baki… | Chicken Casserole | https://hips.hearstapps.com/del.h-cdn.co/assets… | 14 |
| 22 | Extra-virgin olive oil, divided, Juice of 1 lemon, c… | Greek Chicken | https://hips.hearstapps.com/del.h-cdn.co/assets… | 14 |
| 23 | Head cauliflower, lightly beaten, finely grated pa… | Grilled Cheese | https://hips.hearstapps.com/del.h-cdn.co/assets… | 10 |
| 24 | Zucchini, halved lengthwise, extra-virgin olive oil… | Zucchini Boats | https://hips.hearstapps.com/hmg-prod.s3.amaz… | 9 |
| 25 | Lemons, thinly sliced, large salmon fillet, Kosher… | Salmon In Foil | https://hips.hearstapps.com/del.h-cdn.co/assets… | 19 |
| NULL | NULL | NULL | NULL | NULL |



# In detail




```scss
.products-container {
    position: absolute;
    top: 147px;
    /* background-color: lightgray; */
    width: 1111px;
    height: 1080px;
}

.product-card {
    position: relative;
    display: inline-block;
    box-shadow: rgba(0, 0, 0, 0.08) 0px 2px 8px 0px;
    background-color: rgba(255, 255, 255, 0.5);
    width: 350px;
    height: 170px;
    overflow: hidden;
    margin-right: 10px;
    margin-bottom: 10px;
    transition: 0.3s;
    margin: 10px;
    border-radius: 3% / 6%;
    cursor: pointer;
}

.product-card:hover {
    transform: scale(1.1);
}

.photo-container {
    position: absolute;
    height: 110px;
    width: 110px;
    border-radius: 10%;
    margin: 4%;
    /* background-color: lightgray; */
    overflow: hidden;
}

.product-cover {
    position: relative;
    height: 110px;
    width: 110px;
}

.product-card-main {
    position: absolute;
    /* background-color: lightgray; */
    width: 60%;
    height: 73%;
    right: 0;
}
```

```html
<div class="search-box">
  <div class="search-form">
    <input class="search-input searchTerm" type="text" [(ngModel)]="term" />
    <i class="fa fa-search" type="submit"></i>
  </div>
</div>

<div class="best-ones">Best ones so far</div>

<div class="products-container">
  <div
    class="product-card"
    *ngFor="let product of products | searchFilter: term"
  >
    <div class="photo-container">
      <img class="product-cover" [src]="product.pictureUrl" alt="" />
    </div>
    <div class="product-card-main">
      <div class="product-name">{{ product.name }}</div>
      <div class="product-ingredients">{{ product.ingredients }}</div>
    </div>
    <div class="product-card-footer">
      <div class="product-price">{{ product.price }} €</div>
      <button (click)="deleteProduct(product.id)" class="remove-product-button">
        <img
          class="remove-product-button-photo"
          src="/assets/images/remove.png"
          alt=""
        />
      </button>
      <button
        [routerLink]="['/products', product.id]"
        class="edit-product-button"
      >
        <img
          class="edit-product-button-photo"
          src="/assets/images/edit.png"
          alt=""
        />
      </button>
      <button class="add-cart-button">
        <img
          class="add-cart-button-photo"
          src="/assets/images/plus-icon.png"
          alt=""
        />
      </button>
    </div>
  </div>
</div>
```

```ts
@Pipe({
  name: 'searchFilter'
})
export class SearchFilterPipe implements PipeTransform {

  transform(list: any[], filterText: string): any {
    return list ? list.filter(item => item.name.search(new RegExp(filterText, 'i')) > -1) : [];
  }

}
```

Here is the search option in action, along with the pipe used for searching filter.


This is the product card, with the video on the left and the code for that card on the top. This card is created for every product with the help of the *ngFor iteration.

# CRUD operations - Front-end

```
constructor(
  private activatedRoute: ActivatedRoute,
  private httpClient: HttpClient,
  private formBuilder: FormBuilder,
  private router: Router,
  private productService: ProductServiceService) { }

ngOnInit(): void {
  console.log(this.activatedRoute.snapshot.params.id);

  const id = this.activatedRoute.snapshot.params.id;
  this.httpClient.get<AddProduct>('http://localhost:8080/menu/findById?id=' + id).subscribe(response => {
    console.log(response);
    this.myForm.patchValue(response);
  });
}

update() {
  const id = this.activatedRoute.snapshot.params.id;
  this.httpClient.put<AddProduct>('http://localhost:8080/menu/update/' + id, this.myForm.value).subscribe(response => {
    console.log(response);
    this.productService.getAllProducts().subscribe(response => {
        this.products = response;
      });
  });
  this.router.navigate(['/menu']);
}
```

The update method.

```
ngOnInit(): void {
  this.productService.getAllProducts().subscribe(
    response =>this.handleSuccessfulResponse(response),
  );
}

handleSuccessfulResponse(response) {
  this.products=response;
}

deleteProduct(id: number) {
  const confirmed = confirm("Are you sure you want to delete this product?");
  if (confirmed) {
    this.http.delete("http://localhost:8080/menu/delete?id=" + id).subscribe(response => {
      console.log("deleted");
      this.productService.getAllProducts().subscribe(response => {
        this.products = response;
      });
    });
  }
}
```

The delete method.

```
public productName: string;

constructor(private http: HttpClient) { }

public getAllProducts() {
  console.log("test call products");
  return this.http.get<AddProduct[]>('http://localhost:8080/menu/');
}

public createProduct(product: any) {
  return this.http.post<AddProduct>("http://localhost:8080/menu/create", product);
}

public findById(id: any) {
  return this.http.get<AddProduct>("http://localhost:8080/menu/findById");
}

public delete(id: number) {
  const confirmed = confirm("Are you sure you want to delete this product?");
  if (confirmed) {
    this.http.delete("http://localhost:8080/menu/delete/{id}");
  }
}

public findByNameContaining(name: string) {
  return this.http.get<AddProduct[]>("http://localhost:8080/menu/findByNameContaining?name="
  }
}
```

The getAll, create and findBy methods.

```
const routes: Routes = [
{
  path: '',
  component: HomePageComponent,
  pathMatch: 'full'
},
{
  path: 'menu',
  component: MenuPageComponent
},
{
  path: 'our-story',
  component: OurStoryPageComponent
},
{
  path: 'reviews',
  component: ReviewsPageComponent
},
{
  path: 'contact',
  component: ContactPageComponent
},
{
  path: 'log-in',
  component: LogInPageComponent
},
{
  path: 'sign-in',
  component: SignInPageComponent
},
{
  path: 'cart',
  component: CartPageComponent
},
{
  path: 'log-in-after-register',
  component: LogInAfterRegisterComponent
},
{
  path: 'add-product',
  component: AddProductComponent
},
{
  path: 'products/:id',
  component: EditProductComponent
},
{
  path: 'add-review',
  component: AddReviewComponent
}
];
```

The routing module.

```
products: AddProduct[];
public cart: boolean = false;
public filter: boolean = false;


constructor(
  private productService: ProductServiceService) { }

ngOnInit() {
  this.productService.getAllProducts().subscribe(
    response =>this.handleSuccessfulResponse(response),
  );
}

handleSuccessfulResponse(response) {
  this.products=response;
}
```

The method from menu-component.ts that makes viewing products possible.

```
export class AddProduct {
    id: string;
    name: string;
    price: string;
    pictureUrl: string;
    ingredients: string;
}
```

The model class of the “product” object.

The app module class

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    MainContentComponent,
    FooterComponent,
    BottomFooterComponent,
    OurStoryPageComponent,
    MenuPageComponent,
    ReviewsPageComponent,
    ContactPageComponent,
    LogInPageComponent,
    SignInPageComponent,
    NavbarComponent,
    HomePageComponent,
    CartPageComponent,
    LogInAfterRegisterComponent,
    ProductCardComponent,
    AddProductComponent,
    ReviewCardComponent,
    EditProductComponent,
    SearchFilterPipe,
    AddReviewComponent

  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule

  ],
  providers: [
    JwtClientService,
    AlwaysAuthGuard],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# CRUD operations - Back-end

The Controller class responsible for creating endpoints.

```java
@PostMapping("/create")
public ResponseEntity<Product> create(@RequestBody Product product) {
    return ResponseEntity.ok(productService.create(product));
}

@GetMapping("/getAllProducts")
public ResponseEntity<List<Product>> getAllProducts() {
    return ResponseEntity.ok(productService.getAllProducts());
}

@GetMapping("/")
public ResponseEntity<List<Product>> getAllProductsPaginated(
        @RequestParam(defaultValue = "0") Integer pageNumber,
        @RequestParam(defaultValue = "50") Integer pageSize,
        @RequestParam(defaultValue = "id") String sortBy) {
    return ResponseEntity.ok(productService.getAllProductsPaginated(pageNumber, pageSize, sortBy));
}

@GetMapping("/findByName")
public ResponseEntity<Product> findByName(@RequestParam String name) {
    return ResponseEntity.ok(productService.findByName(name).orElseThrow(() -> new ProductNotFoundException("Product with name <" + name + "> not found")));
}

@GetMapping("/findById")
public ResponseEntity<Product> findById(@RequestParam Integer id) {
    return ResponseEntity.ok(productService.findById(id).orElseThrow(() -> new ProductNotFoundException("Product with id <" + id + "> not found")));
}

@PutMapping("/update/{id}")
public ResponseEntity<Product> update(
        @RequestBody Product product,
        @PathVariable("id") Integer id) {

    Optional<Product> productData = productService.findById(id);

    if (productData.isPresent()) {
        Product productToBeChanged = productData.get();
        productToBeChanged.setName(product.getName());
        productToBeChanged.setIngredients(product.getIngredients());
        productToBeChanged.setPrice(product.getPrice());
        productToBeChanged.setPictureUrl(product.getPictureUrl());
        return new ResponseEntity<>(productService.save(productToBeChanged), HttpStatus.OK);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}
```

The model according to the object “product”.

```java
@Data
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Table(name = "PRODUCT_TBL")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @Column
    private String name;

    @Column
    private Double price;

    @Column
    private String pictureUrl;

    @Column
    private String ingredients;


}
```

```java
public interface ProductRepository extends PagingAndSortingRepository<Product, Integer> {

    Optional<Product> findByName(String name);

    List<Product> findByNameContaining(String name);
}
```

The repository.

```java
public interface ProductService {

    Product create(Product product);

    List<Product> getAllProducts();

    List<Product> getAllProductsPaginated(Integer pageNumber, Integer pageSize, String sortBy);

    Optional<Product> findByName(String name);

    Optional<Product> findById(Integer id);

    Product save(Product product);

    Product update(Product product, Integer id);

    void delete(Integer id);

    List<Product> getByNameContaining(String name);
}
```

```java
@Override
public Product create(Product product) {
    return productRepository.save(product);
}

@Override
public List<Product> getAllProducts() {
    return (List<Product>) productRepository.findAll();
}

@Override
public List<Product> getAllProductsPaginated(Integer pageNumber, Integer pageSize, String sortBy) {
    Pageable pageable = PageRequest.of(pageNumber, pageSize, Sort.by(sortBy));
    Page<Product> productPage = productRepository.findAll(pageable);
    return productPage.getContent();
}

@Override
public Optional<Product> findByName(String name) { return productRepository.findByName(name); }

@Override
public Optional<Product> findById(Integer id) { return productRepository.findById(id); }

@Override
public Product save(Product product) { return productRepository.save(product); }

@Override
public Product update(Product product, Integer id) { return productRepository.save(product); }

@Override
public void delete(Integer id) {
    Optional<Product> product = productRepository.findById(id);

    if (product.isPresent()) {
        productRepository.deleteById(id);
    }else {
        throw new ProductNotFoundException("Product with id " + id + " not found.");
    }
}

@Override
public List<Product> getByNameContaining(String name) {
    return productRepository.findByNameContaining(name);
}
}
```

# The Service interface and class for the product object

# 03

# Our Story

# The Our Story Page


```
    </div>
    <div class="founders-description-container">
      <div class="f-d-1">
        James Smith realized his culinary vision at his first eponymous
        restaurant, in the very heart of downtown Vancouver. At Hawksworth
        Restaurant, which opened in 2011, he has created the ideal setting to
        deliver compelling contemporary Canadian cuisine.
      </div>
      <div class="f-d-2">
        After prestigious stints on the Miami, San Fran and Toronto culinary
        scenes, Chef David settled in Vancouver at Hawksworth as Senior Sous
        Chef before taking the helm at Botanist. Under his direction, Botanist
        has been named one of Canada's 50 Best Restaurants.
      </div>
      <div class="f-d-3">
        Originally hailing from Paris, France, this globetrotter has been in
        the hospitality industry for over a decade, working alongside
        internationally celebrated chefs including Jean Georges, Nobu
        Matsuhisa and Justin Quek.
      </div>
    </div>
  </div>
</div>
<div class="our-story-footer">
  <img class="footer-img" src="/assets/images/our-story-footer.jpg" alt="" />

  <div class="bottom-footer">
    <a href="#"><div class="terms_and_conditions">terms&conditions</div></a>
    <div class="footer_logo">
      <a routerLink=""
        ><img class="logo" src="assets/images/white_logo.png" alt=""
      /></a>
    </div>
    <div class="footer_social_media">
      <div class="f_logo">
        <a href="#"><img src="assets/images/f_logo.png" alt="" /></a>
      </div>
      <div class="i_logo">
        <a href="#"><img src="assets/images/i_logo.png" alt="" /></a>
      </div>
      <div class="t_logo">
        <a href="#"><img src="assets/images/t_logo.png" alt="" /></a>
      </div>
    </div>
    <div class="up">
      <a href="#top"><img src="assets/images/up.png" alt="" /></a>
    </div>
  </div>
</div>
</body>
```

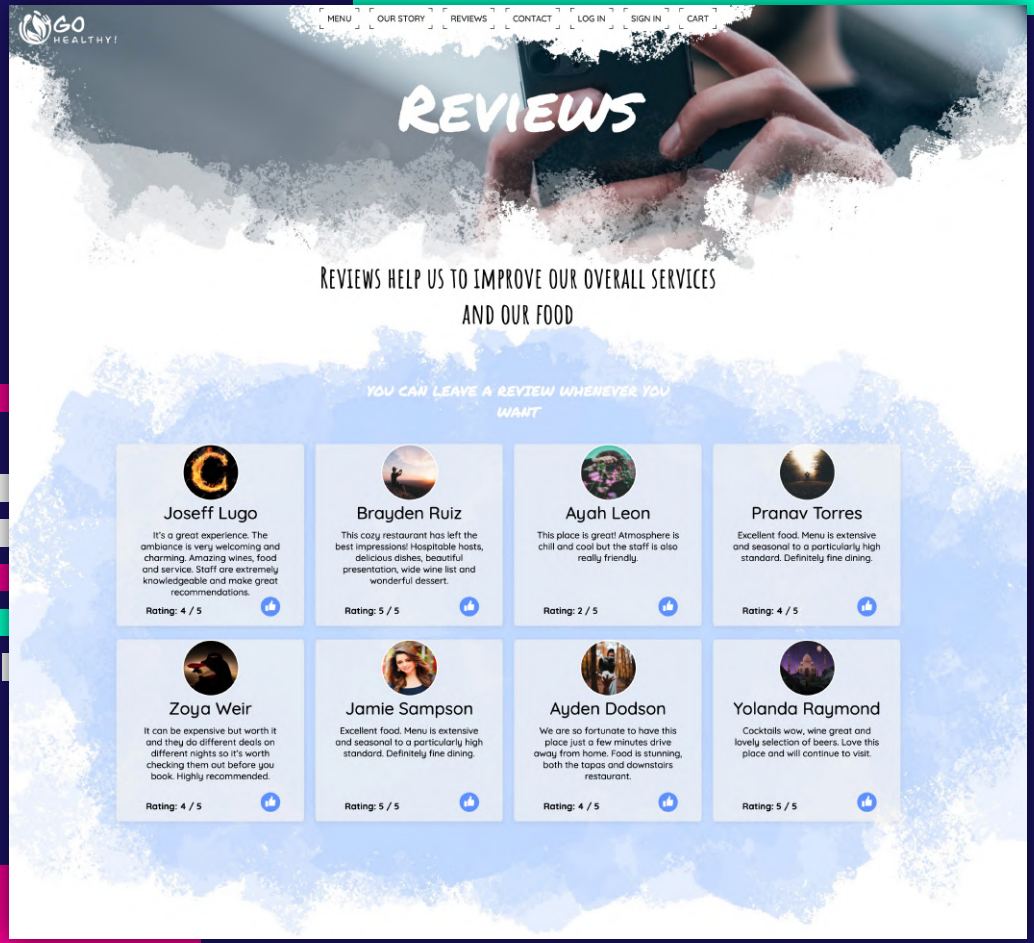
The Our Story page contains information like who are the founders of the restaurant, or what is the goal they want to achieve.

```
.founders-name-container {
    position: absolute;
    /* background-color: grey; */
    width: 90%;
    top: 20%;
    height: 10%;
    left: 5%;
    display: flex;
    justify-content: space-between;
}

.f1, .f2, .f3 {
    width: 20%;
    height: 100%;
    /* background-color: darkgray; */
    font-family: 'Amatic SC', cursive;
    font-size: 40px;
    text-align: center;
}

.founders-description-container {
    position: absolute;
    /* background-color: grey; */
    width: 100%;
    height: 60%;
    bottom: 0;
    display: flex;
    justify-content: space-between;
}

.f-d-1, .f-d-2, .f-d-3 {
    width: 30%;
    height: 100%;
    left: 3%;
    /* background-color: darkgray; */
    font-family: 'Quicksand', sans-serif;
    text-align: center;
    font-size: 30px;
    font-weight: 300;
}

.our-story-footer {
    position: absolute;
    background-color: lightgray;
    width: 100vw;
    top: 180%;
}

.footer-img {
    width: 100%;
}
```

Here are a part of the HTML and CSS of this page.

# 04

# Reviews



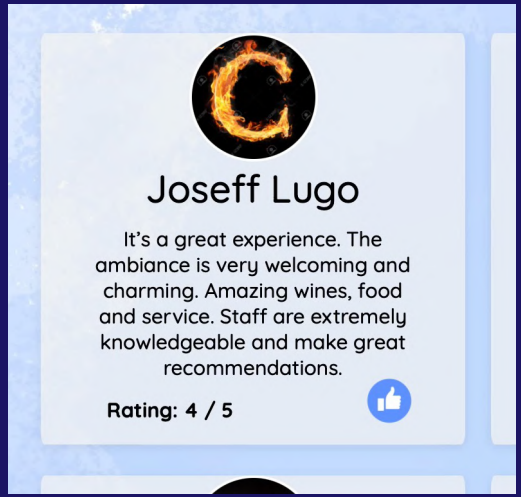

# The Reviews Component - Front-end

The Reviews page serves for customers to leave their feedback. Like the “Product” object, in the Reviews page are implemented the CRUD methods.


Here are some details from the Reviews page, down it's a video of the review card Design, and on the left is displayed the HTML code from the same card.

# In detail

```html
<div class="reviews-container">
    <div class="review-card" *ngFor="let review of reviews">
        <div class="photo-container">
            <img class="review-cover" [src]="review.pictureUrl" alt="">
        </div>
        <div class="review-card-main">
            <div class="review-name">{{ review.name }}</div>
            <div class="review-description">{{ review.description }}</div>
        </div>
        <div class="review-card-footer">
            <div class="review-rating">Rating: {{ review.rating }} / 5</div>
            <button class="review-like-button">
                <img class="review-like-photo" src="/assets/images/reviews-like-button.png" alt="">
            </button>
        </div>
    </div>
</div>
```




```css
.review-description {
  position: absolute;
  /* background-color: darkgray; */
  font-family: "Quicksand", sans-serif;
  width: 100%;
  height: 70%;
  bottom: 0;
  text-align: center;
  /* line-height: 100px; */
}

.review-card-footer {
  position: absolute;
  width: 80%;
  height: 15%;
  /* background-color: lightgray; */
  right: 10%;
  bottom: 1%;
}

.review-like-button {
  position: absolute;
  height: 30px;
  width: 30px;
  border: 0;
  cursor: pointer;
  right: 0;
  bottom: 10%;
  background-color: transparent;

}

/* .review-like-button:hover {
    transform: scale(1.2);
} */

.review-like-photo {
  position: absolute;
  height: 50px;
  width: 50px;
  bottom: 1px;
  right: 4%;
  /* transition: all 1s ease-in-out, height 0ms; */
  transition: transform 1s;
}

.review-like-photo:hover {
    /* transform: scale(1.2) rotate(-30deg) ; */
    transform: scale(1.2) rotate(1080deg);

}
```

# Review Angular Service, the create new Review and get all Reviews method

```
  constructor(private http: HttpClient) { }

  public getAllReviews() {
    console.log("test call reviews");
    return this.http.get<AddReview[]>("http://localhost:8080/reviews/getAllReviews");
  }

  public createReview(review: any) {
    return this.http.post<AddReview>("http://localhost:8080/reviews/create", review);
  }
}
```

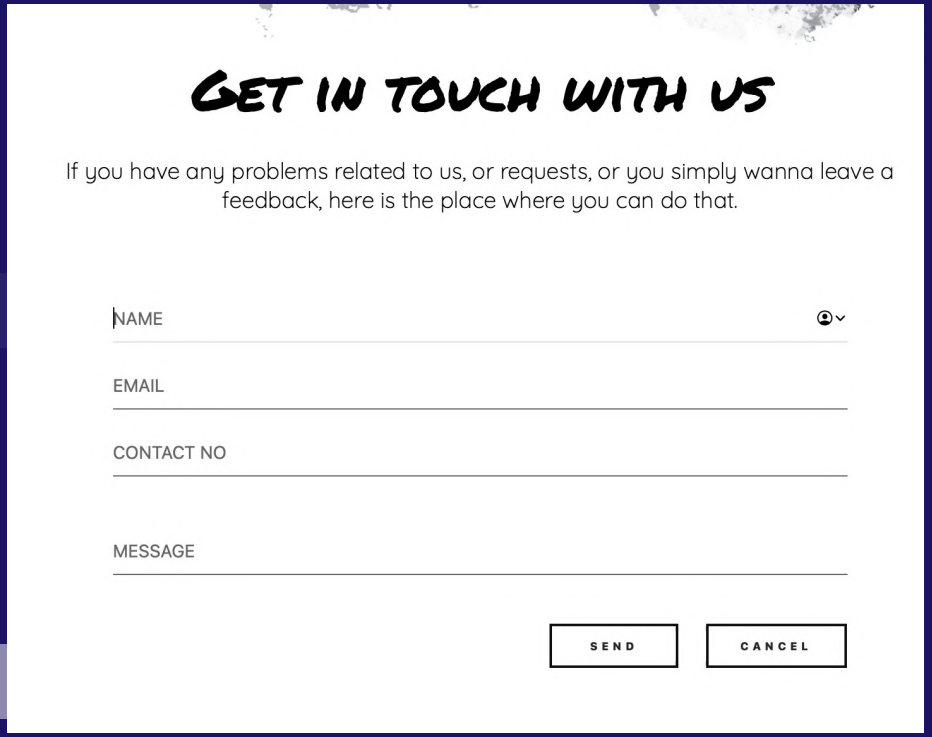
```
onSubmit() {
  this.reviewService.createReview(this.review).subscribe(data => {
    this.response = data;
  });

  this.router.navigate(['/reviews']);
}
```

```
export class AddReview {

    id: string;
    name: string;
    pictureUrl: string;
    description: string;
    rating: string;
}
```


The Contact page has all the information about the location, and an embedded map for easy orientation.

Besides that, on the left, it's a form for the client to contact the restaurant, and when the user clicks on send, it will pop up a page for the mail to be sent.

# The Contact Component - Front-end




Here is a video of the form and the animated buttons, and, at the opposite Side, It's the HTML code for de location and map part, and the CSS code for the Send button.

```css
/* Send Button */

.send_button {
    background: transparent;
    position: absolute;
    top: 110%;
    right: 23%;
    border: 2px solid #111;
    padding: 10px 30px;
    overflow: hidden;
}

.send_button:hover:before {
    opacity: 1;
    transform: translate(0,0);
}

.send_button::before {
    content: attr(data-hover);
    position: absolute;
    top: 1.1em;
    left: 0.2em;
    width: 100%;
    text-transform: uppercase;
    letter-spacing: 3px;
    font-weight: 800;
    font-size: .8em;
    opacity: 0;
    transform: translate(-100%,0);
    transition: all .3s ease-in-out;
}

.send_button:hover div {
    opacity: 0;
    transform: translate(100%, 0);
}

.send_button div{
    text-transform: uppercase;
    letter-spacing: 3px;
    font-weight: 800;
    font-size: .8em;
    transition: all .3s ease-in-out;
}
```

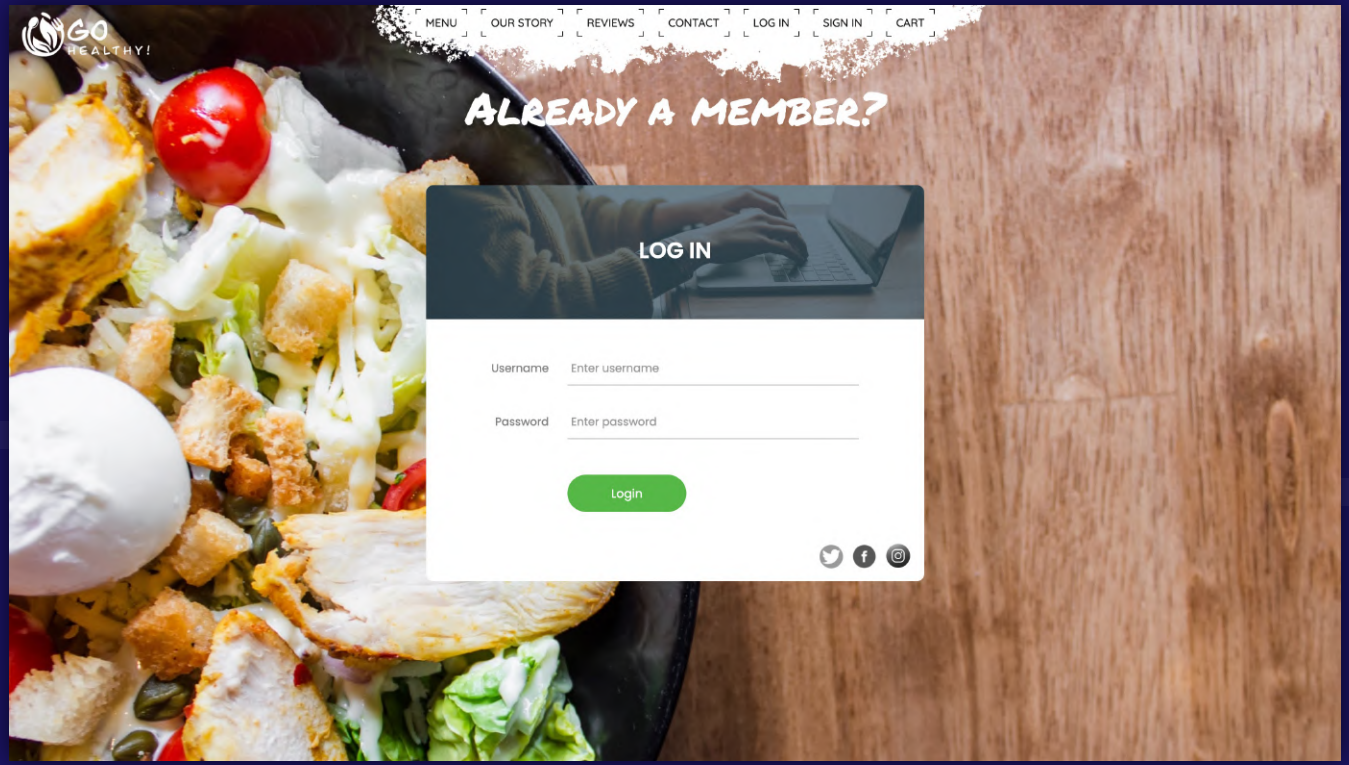
# In detail

```html
<div class="contact_break_line"></div>
<div class="general_informations">
  <div class="general_informations_title">General informations</div>
  <div class="informations">
    <div class="mail">Mail: gohealthy@yahoo.com</div>
    <div class="phone">Phone: +1-555-8099-004</div>
    <div class="company_name">GO HEALTHY LLC</div>
    <div class="established">Established in 2014</div>
    <div class="where">
      <div class="state">California, San Jose</div>
      <div class="address">3877 Driftwood Road</div>
      <div class="zip">95129</div>
    </div>
  </div>
  <div class="map">
    <iframe
      src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d3173.858560
      width="682"
      height="265"
      style="border: 0"
      allowfullscreen=""
      loading="lazy"
    ></iframe>
  </div>
</div>
<div class="contact_footer">
  <div class="footer_photo">
    <img src="assets/images/footer_photo.jpg" alt="" />
  </div>
  <div class="footer_links">
    <a href="#"><div class="terms_and_conditions">terms&conditions</div></a>
    <div class="footer_logo">
      <a router
        ><img class="logo" src="assets/images/white_logo.png" alt=""
        /></a>
    </div>
    <div class="footer_social_media">
      <div class="f_logo">
        <a href="#"><img src="assets/images/f_logo.png" alt="" /></a>
      </div>
      <div class="i_logo">
        <a routerLink=""><img src="assets/images/i_logo.png" alt="" /></a>
      </div>
      <div class="t_logo">
        <a href="#"><img src="assets/images/t_logo.png" alt="" /></a>
      </div>
    </div>
  </div>
</div>
```

# 06

# Log In

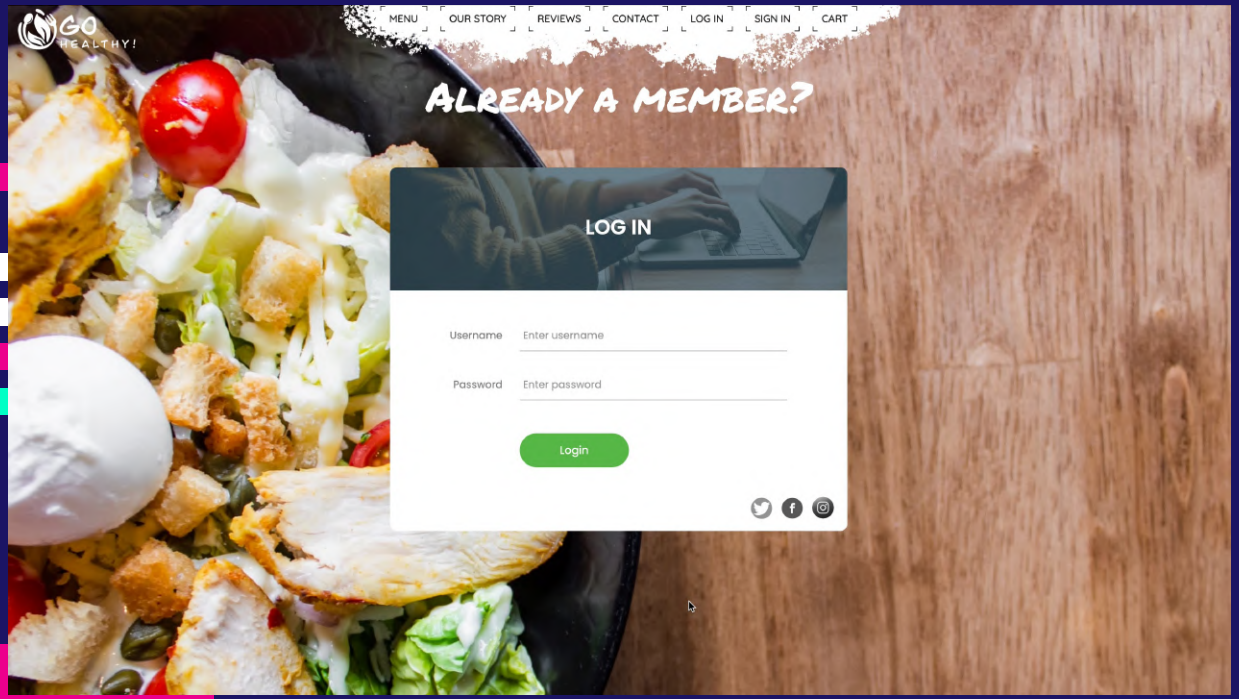
# The Log In Component – Front-end




The authentication on this project was done with the JWT Token, which is a more secure way to log in.
When you submit a username and a password, and if they are correct, it will be generated an unique token formed by 3 parts.
With that token, the app will know if the client is already registered and in that case it will redirect him to the Home page, and if not, he would have to register first, and then log in.

# In detail

Here is a video of that form used for login, and on the right side, it is the HTML code for that form.



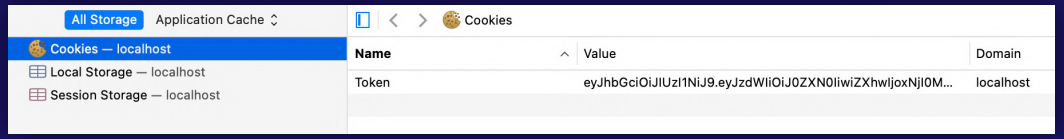

```
<div class="limiter">
  <div class="container-login100">
    <div class="wrap-login100">
      <div
        class="login100-form-title"
        style="background-image: url(/assets/images/login2.jpg)"
      >
        <span class="login100-form-title-1"> Log In </span>
      </div>

      <form class="login100-form validate-form" role="form">
        <div
          class="wrap-input100 validate-input m-b-26"
          data-validate="Username is required"
        >
          <span class="label-input100">Username</span>
          <input
            class="input100"
            [(ngModel)]="username"
            type="text"
            name="username"
            placeholder="Enter username"
          />
          <span class="focus-input100"></span>
        </div>

        <div
          class="wrap-input100 validate-input m-b-18"
          data-validate="Password is required"
        >
          <span class="label-input100">Password</span>
          <input
            class="input100"
            [(ngModel)]="password"
            type="password"
            name="pass"
            placeholder="Enter password"
          />
          <span class="focus-input100"></span>
        </div>

        <div class="container-login100-form-btn">
          <button class="login100-form-btn" (click)="logIn()">Login</button>
        </div>
      </form>
```

# The entity, the login method, the service for the user, and the generate token request.

```
export class UserSignIn {
    id: string;
    username: string;
    email: string;
    password: string;
}
```

```
export class JwtClientService {

  private httpOptions = {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' }),
    responseType: "text" as "json",
    withCredentials: true,
    observe: 'response' as 'response'
  };

  constructor(private http:HttpClient) { }

  public generateToken(request) {
    return this.http.post<any>("http://localhost:8080/authenticate", request, this.httpOptions);
  }

  public welcome(token) {
    let tokenStr = 'Bearer ' + token;
    const headers = new HttpHeaders().set("Authorization", tokenStr);
    return this.http.get("http://localhost:8080/", {headers, responseType: 'text' as 'json'});
  }

  public register(user: any) {

    return this.http.post<UserSignIn>("http://localhost:8080/register", user);
  }
}
```

A good thing to be mentioned is the fact that after the token got generated, il will be stored in an HTTP-only cookie, to prevent access to him.




```
export class LogInPageComponent implements OnInit {

  token: String;
  username: String;
  password: String;
  user: any;

  @Input() titleName: string = 'Already\na\nmember?';

  constructor(public authService: JwtClientService, public router: Router) { }

  ngOnInit(): void {

  }

  public logIn() {
    this.user = {
      "username": this.username,
      "password": this.password
    }
    let response = this.authService.generateToken(this.user);
    response.subscribe(data => {
        if (data.body == 'true') {
          this.router.navigate(["/"]);
        }
    },(err) => {
      console.log(err.status);
    })
  }

}
```

# The Log In Component - Back-end

The Controller Advisor class provides the method that returns an exception if the product is not found.

```java
@ControllerAdvice
public class ControllerAdvisor extends ResponseEntityExceptionHandler {

    @ExceptionHandler(ProductNotFoundException.class)
    public ResponseEntity<Object> handleProductNoyFoundException(ProductNotFoundException exception) {
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("timestamp", LocalDateTime.now());
        body.put("message", exception.getLocalizedMessage());
        return new ResponseEntity<>(body, HttpStatus.NOT_FOUND);
    }
}
```

```java
@SpringBootApplication
@EntityScan("com.project.goHealthy.entity")
@EnableJpaRepositories("com.project.goHealthy.repository")
@ComponentScan("com.project.goHealthy")
@Import({SecurityConfig.class})
public class AppConfig {
}
```

The appConfig class for the application, that scan all the entities.

```java
@SpringBootApplication
@CrossOrigin(origins = "*")
public class GoHealthyApplication {

    private UserRepository userRepository;

    private ProductRepository productRepository;

    @Autowired
    public GoHealthyApplication(UserRepository userRepository, ProductRepository productRepository) {
        this.userRepository = userRepository;
        this.productRepository = productRepository;
    }

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return addCorsMappings(registry) -> {
                registry.addMapping( pathPattern: "/*")
                        .allowedHeaders("*")
                        .allowedMethods("*")
                        .allowedOriginPatterns("*")
                        .allowCredentials(true);
        };
    }

    public static void main(String[] args) { SpringApplication.run(GoHealthyApplication.class, args); }
}
```

The Spring application class, that run the code.

The entity for the "User" object. Besides that, with the help of this class, it is created the table in the MySql Database.

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
@Setter
@Getter
@Entity
@Table(name = "USER_TBL")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column
    private String username;

    @Column
    private String password;

    @Column
    private String email;
}
```

```java
public interface UserRepository extends JpaRepository<User, Integer> {

    User findByUsername(String username);
}
```

The User Repository that extends the Jpa Repository.

# The Log In Component - Back-end - User Configuration

The Custom Details Service class, responsible for the loadUserByUsername method.

```java
@Service
public class CustomUserDetailsService implements UserDetailsService {

    private final UserRepository repository;

    @Autowired
    public CustomUserDetailsService(UserRepository repository) { this.repository = repository; }

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = repository.findByUsername(username);
        return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), new ArrayList<>());
    }
}
```

# The Log In Component – Back-end – Security

```java
@PostMapping("/authenticate")
public boolean generateToken(@RequestBody AuthRequest authRequest, HttpServletResponse response) throws Exception {
    try {
        authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(authRequest.getUsername(), authRequest.getPassword())
        );
    }catch (Exception ex) {
        throw new Exception("invalid username/password");
    }
    String token = jwtUtil.generateToken(authRequest.getUsername());
    Cookie cookie = new Cookie( name: "Token", token);
    cookie.setHttpOnly(true);
    cookie.setMaxAge(-1);
    cookie.setPath("/");
    cookie.setSecure(false);
      response.addHeader("Access-Control-Allow-Headers", "http://localhost:4200");
    response.addCookie(cookie);
    return true;
}
```

The authenticate method from User Controller, which verifies in the first part if the credentials are correct, and then authenticate the user, and at the end of the method it moves the generated token to an HTTP-only cookie.

The SecurityConfig class overrides some of the WebSecurityConfigurerAdapter methods. Besides that, it uses the method passwordEncoder to save passwords directly encoded in the database. This way, not even the administrator will be able to see the passwords.

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private DataSource dataSource;

    @Autowired
    private CustomUserDetailsService userDetailsService;

    @Autowired
    private JwtFilter jwtFilter;

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
        auth.jdbcAuthentication().dataSource(dataSource);
    }

    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }

    @Bean(name = BeanIds.AUTHENTICATION_MANAGER)
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.cors().disable();
        http.csrf().disable().authorizeRequests().antMatchers( ...antPatterns: "/**") ExpressionUrlAuthorizationConfigurer<H>.AuthorizedUrl
                .permitAll().antMatchers(HttpMethod.OPTIONS, ...antPatterns: "/**")
                .permitAll().anyRequest().authenticated() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
                .and().exceptionHandling().and().sessionManagement() SessionManagementConfigurer<HttpSecurity>
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
        http.addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
    }
}
```

```java
private String createToken(Map<String, Object> claims, String subject) {
    return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
            .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
            .signWith(SignatureAlgorithm.HS256, secret).compact();
}

public Boolean validateToken (String token, UserDetails userDetails) {
    final String username = extractUserName(token);
    return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
}
```

The createToken and the validateToken methods from JwtUtil.

The JwtFilter is a custom filter that was made to run first, compared to the already implemented Spring filters.

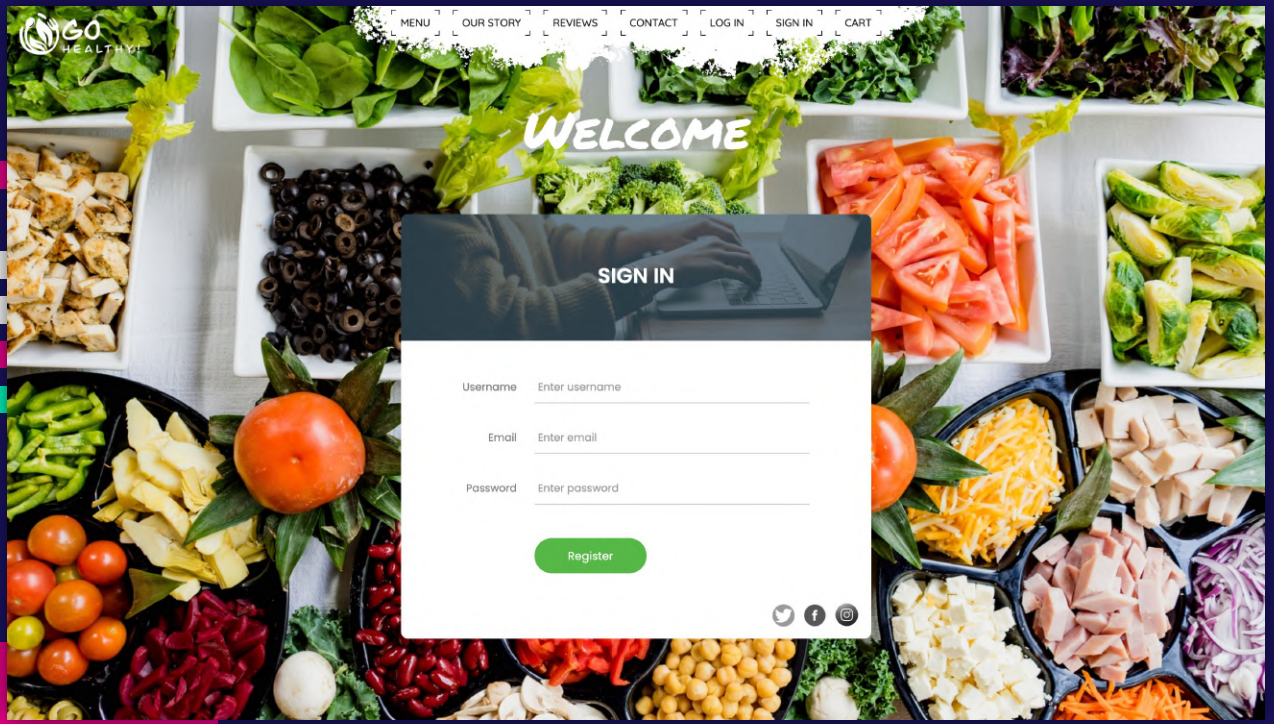
# The LogIn Component - Back-end - JWT Token

```java
@Component
public class JwtFilter extends OncePerRequestFilter {

    private final JwtUtil jwtUtil;

    private final CustomUserDetailsService service;

    @Autowired
    public JwtFilter(JwtUtil jwtUtil, CustomUserDetailsService service) {
        this.jwtUtil = jwtUtil;
        this.service = service;
    }

    @Override
    protected void doFilterInternal(
            HttpServletRequest httpServletRequest,
            HttpServletResponse httpServletResponse,
            FilterChain filterChain) throws ServletException, IOException {
        String token = null;
        String userName = null;
        String authorizationHeader = httpServletRequest.getHeader( s: "Authorization");
        if (authorizationHeader != null && authorizationHeader.startsWith("Bearer ")) {
            token = authorizationHeader.substring(7);
            userName = jwtUtil.extractUserName(token);
        }

        if (userName != null && SecurityContextHolder.getContext().getAuthentication() == null) {
            UserDetails userDetails = service.loadUserByUsername(userName);
            if (jwtUtil.validateToken(token, userDetails)) {


                UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(
                        userDetails, credentials: null,
                        userDetails.getAuthorities());
                usernamePasswordAuthenticationToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(httpServletRequest));
                SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);
            }
        }
        filterChain.doFilter(httpServletRequest, httpServletResponse);
    }
}
```

# 07

# Sign In

# The Sign In Component - Front-end




The Sign In page provides the user with a method to make an account on the site. The input values are validated and if all is good, the request to the back-end is made and the new user is created in the database.

# The Sign In Component Back-end - Register

```java
@PostMapping("/register")
public void register(@RequestBody User user) {
    System.out.println("Register accessed");
    user.setPassword(passwordEncoder.encode(user.getPassword()));
    System.out.println(user.getPassword());
    userRepository.save(user);
}
}
```

The register method which creates a new user.

| id | email | password | username |
|---|---|---|---|
| 1 | codrut@yahoo.com | $2a$10$KfRcMJaEp1kQmKRKCzZcG.4GbXM… | codrut |
| 2 | hallo@yahoo.com | $2a$10$rR0TTqOmixGEYRqPqvEw0uycL4w87… | hallo |
| 3 | test7@yahoo.com | $2a$10$KwDiuSK9grdClsqb8yqKzeSg6YVBOk… | test7 |
| 4 | test2@yahoo.com | $2a$10$wIFb8Eh7QjhzELtNC3JWHeiIcdBebKF… | test2 |
| 5 | test3@yahoo.com | $2a$10$PTB7qX2ZYLUQlw11H3F5j.tanL10JLj… | test3 |
| 6 | test4@yahoo.com | $2a$10$GbQuXVlSTue1wiBDXr.owOoQskfxCv… | test4 |
| 7 | test5@yahoo.com | $2a$10$b.0x.sAjy2U2VmnHf0vSHujRCK4w9k3… | test5 |
| 8 | test6@yahoo.com | $2a$10$9ntLppKbWt8VrD79N1DzQeGQtH2U2… | test6 |
| 9 | test@yahoo.om | $2a$10$9QglvhnI/6DpHWAsi0sH9xOk7pT/ruFd… | test |
| 10 | test9@yahoo.com | $2a$10$gvQDuXTroQDJz2uIzQH7xe.3a6C2pQ… | test9 |
| 11 | test1@yahoo.com | $2a$10$MjfD6zhzFwtFlvDHlp/oheo/rhJmPMiqt… | test1 |
| NULL | NULL | NULL | NULL |

This is the database table with the encoded password that comes from the back-end.

An example of email validate method.

```typescript
export class SignInPageComponent {

  employeeForm: FormGroup;
  user: UserSignIn;
  response: any;

  constructor(
    public authSevice: JwtClientService,
    private router: Router) {
    this.user = new UserSignIn();
  }

  ngOnInit() {
    this.employeeForm = new FormGroup({
      username: new FormControl(),
      email: new FormControl()
    });

    var emailEl = document.getElementById("email");
    emailEl.addEventListener("blur", this.emailIsValid, false);
  }

  onSubmit() {
    this.authSevice.register(this.user).subscribe(data => {
      this.response = data;
    });

    if (this.emailIsValid() == true) {
      this.router.navigate(['/log-in-after-register']);
    }
  }

  public emailIsValid() {
    let email = this.user.email;
    let regx = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/;

    if (regx.test(email)) {
      return true;

    } else {
      alert("Bad");
      // console.log("Bad");
      return false;
    }
  }
}
```


Thank you for your attention!