Report

Introduction :

This report highlights the development project of a web application for the airline Tunisair. The objective of this application is to facilitate and optimize the management of flights, aircrew and airline catering partners. Additionally, the project incorporates an artificial intelligence module to predict flight delays, contributing to better planning and a smoother travel experience for passengers.

The first part of this report presents an overview of the project, highlighting key features of the application, such as flight planning, crew management, catering partner management and delay prediction. We also look at the technologies used for the development of the application, including Angular and Spring Boot for the front end and backend respectively, as well as the use of JWT to guarantee data security and the Catboost Regresor model for our AI model.

The second part of the report focuses on the architecture and design of the application. We will detail the different components of the application, their interaction and their integration. Additionally, we discuss the design choices, data models, and workflows that underpin the application.

Finally, the report will conclude with a summary of the results obtained, the advantages of the application developed for Tunisair and future prospects

Analyse of needs :

Our application is intended mainly for 3 types of users according to their roles:

Admin

Moderator

User

Admin

- Manages the list of users (Admin, Moderator, User)
- Manages the list of pilots
- Manages the list of co-pilots
- Manages the staff list
- Manages catering businesses
- Manages flights
- Request predictions by the artificial intelligence algorithm

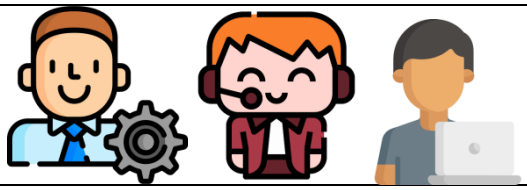
Moderator

- Manages the list of pilots
- Manages the list of co-pilots
- Manages the staff list
- Manages catering businesses
- Manages flights
- Request predictions by the artificial intelligence algorithm

User

- Consult the list of pilots
- Consult the list of co-pilots
- Consult the list of staff
- Consult catering companies
- Check flights
- Request predictions by the artificial intelligence algorithm

Note: All access and authorizations for the 3 types of users are managed by the restrictions by JWT of Spring Boot, and also by the restrictions in the Front end side where each user will only have access to the functionalities of which he has the right of access.



A user is characterized by:

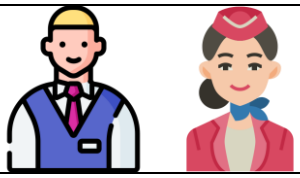
-Id
-Name
-Email
-Code
-Job Title
-Roles

A Piolte is characterized by:

-Id
-Name
- Family Name
- Email
-Password
-Date of Birth
-Phone
-Gender

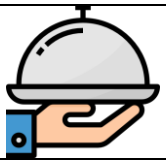
A Copilote is characterized by:

- Id
-Name
- Family Name
- Email
-Password
-Date of Birth
-Phone
-Gender



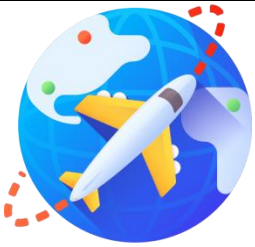
A Staff is characterized by:

- Id
-Name
- Family Name
- Email
-Password
-Date of Birth
-Phone
-Role



A catering company is characterized by:

-Id
-Name
-Country
-Price
-Menus



A Flight is characterized by:

-Id
-Date de depart
-city de depart
-city d'arrivée
-Type de voyage
-Pilote
-Copilote
-Staffs
-Catering companies

Design and Architecture Back End, Front End and the AI model:

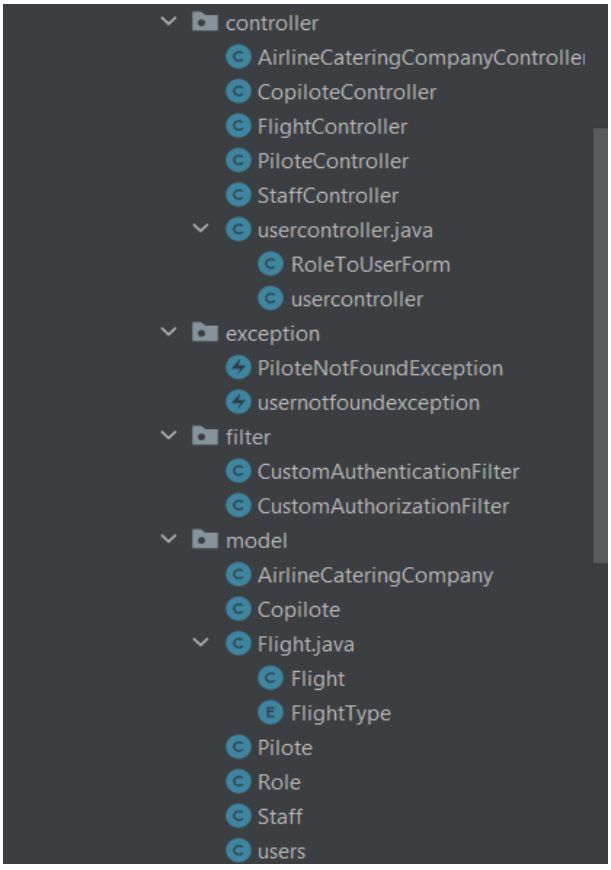
Back End

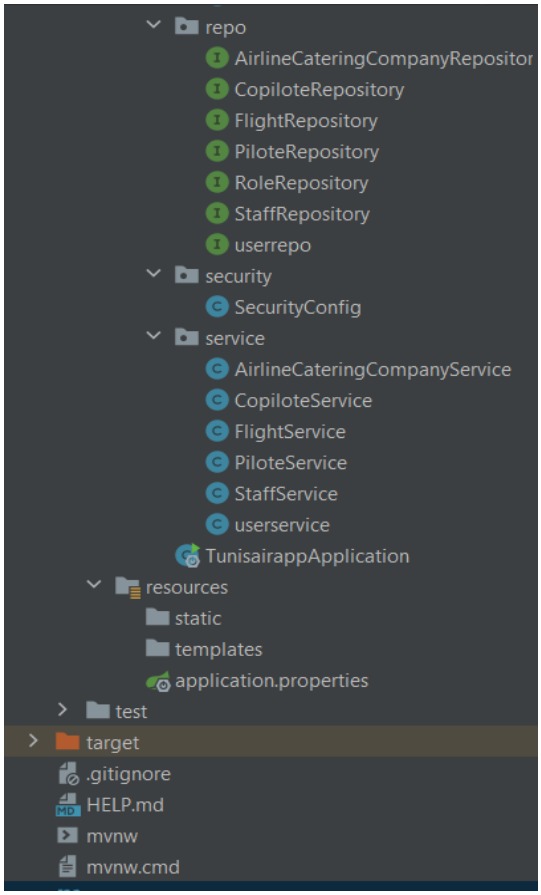
The necessary dependencies for our Spring Boot project are:

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <!-- Spring Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL Driver -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.23</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
```

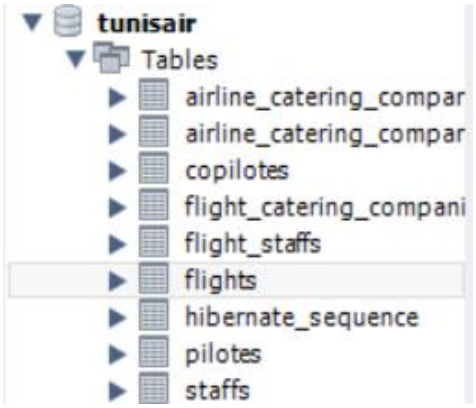
```
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>com.auth0</groupId>
        <artifactId>java-jwt</artifactId>
        <version>4.4.0</version>
    </dependency>
</dependencies>
```

The general breakdown of our project will be as follows:





The models contain the classes that represent the tables of our Mysql database managed by Mysql WorkBench:



With the following configuration for Mysql in the application properties file:

```
#MySQL Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/tunisair
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

```
@Entity
@Table(name = "users") @NoArgsConstructor
public class users implements Serializable {
    @Id
    @GeneratedValue( generator = "uuid2" )
    @GenericGenerator( name = "uuid2", strategy = "uuid2" )
    @Column( name = "id", columnDefinition = "BINARY(16)" )
    private UUID id;
    private String name;
    private String email;
    @Column(nullable = false)
    private String code;
    private String jobtitle;
    @ManyToMany(fetch = EAGER)
    private Collection<Role> roles = new ArrayList<>();
```

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Role {
    @Id
    @GeneratedValue(strategy = AUTO)
    private Long id;
    private String name;
}
```

```
@Entity
@Table(name = "copilotes")
public class Copilote implements Serializable {

    @Id
    @GeneratedValue( generator = "uuid2" )
    @GenericGenerator( name = "uuid2", strategy = "uuid2" )
    @Column( name = "id", columnDefinition = "BINARY(16)" )
    private UUID id;

    private String name;

    @Column(name = "family_name")
    private String familyName;

    private String email;

    private String password;

    @Column(name = "date_of_birth")
    private Date dateOfBirth;

    private String phone;
    private String gender;
```

```
@Entity
@Table(name = "pilotes")
public class Pilote implements Serializable {

    @Id
    @GeneratedValue( generator = "uuid2" )
    @GenericGenerator( name = "uuid2", strategy = "uuid2" )
    @Column( name = "id", columnDefinition = "BINARY(16)" )
    private UUID id;

    private String name;

    @Column(name = "family_name")
    private String familyName;

    private String email;

    private String password;

    @Column(name = "date_of_birth")
    private Date dateOfBirth;

    private String phone;
    private String gender;
```

```java
@Entity
@Table(name = "staffs")
public class Staff implements Serializable {

    @Id
    @GeneratedValue( generator = "uuid2" )
    @GenericGenerator( name = "uuid2", strategy = "uuid2" )
    @Column( name = "id", columnDefinition = "BINARY(16)" )
    private UUID id;

    private String name;

    @Column(name = "family_name")
    private String familyName;

    private String email;

    private String password;

    @Column(name = "date_of_birth")
    private Date dateOfBirth;

    private String phone;
    private String role;
```

```java
@Entity
@Table(name = "airline_catering_companies")
public class AirlineCateringCompany implements Serializable {

    @Id
    @GeneratedValue(generator = "uuid2")
    @GenericGenerator(name = "uuid2", strategy = "uuid2")
    @Column(name = "id", columnDefinition = "BINARY(16)")
    private UUID id;

    private String name;
    private String country;
    private int price;

    @ElementCollection
    private List<String> menu;
```

```java
    private String departureCity;

    @Column(name = "destination_city")
    private String destinationCity;

    @Enumerated(EnumType.STRING)
    private FlightType type;

    @ManyToOne
    @JoinColumn(name = "pilot_id")
    private Pilote pilot;

    @ManyToOne
    @JoinColumn(name = "copilot_id")
    private Copilote copilot;

    @ManyToMany
    @JoinTable(
            name = "flight_staffs",
            joinColumns = @JoinColumn(name = "flight_id"),
            inverseJoinColumns = @JoinColumn(name = "staff_id")
    )
    private List<Staff> staffs;

    @ManyToMany
    @JoinTable(
            name = "flight_catering_companies",
            joinColumns = @JoinColumn(name = "flight_id"),
            inverseJoinColumns = @JoinColumn(name = "catering_company_id")
    )
    private List<AirlineCateringCompany> cateringCompanies;
```

These classes are linked to repositories interfaces to define methods like findbyid(), findbyemail(),…

These classes are connected to controllers and services for back end processing, and here is the example of the services and controllers of the flight class:

Services

```java
@Service
public class FlightService {

    private final FlightRepository flightRepository;

    @Autowired
    public FlightService(FlightRepository flightRepository) {
        this.flightRepository = flightRepository;
    }

    public List<Flight> getAllFlights() {
```

```
        return flightRepository.findAll();
    }

    public Optional<Flight> getFlightById(UUID id) {
        return flightRepository.findById(id);
    }

    public Flight saveFlight(Flight flight) {
        return flightRepository.save(flight);
    }

    public void deleteFlight(UUID id) {
        flightRepository.deleteById(id);
    }
}
```

Controllers

```
@RestController
@RequestMapping("/flights")
public class FlightController {

    private final FlightService flightService;

    @Autowired
    public FlightController(FlightService flightService) {
        this.flightService = flightService;
    }

    @GetMapping("/all")
    public ResponseEntity<List<Flight>> getAllFlights() {
        List<Flight> flights = flightService.getAllFlights();
        return new ResponseEntity<>(flights, HttpStatus.OK);
    }

    @GetMapping("/find/{id}")
    public ResponseEntity<Flight> getFlightById(@PathVariable UUID id) {
        Optional<Flight> flight = flightService.getFlightById(id);
        return flight.map(value -> new ResponseEntity<>(value,
HttpStatus.OK))
                .orElseGet(() -> new
ResponseEntity<>(HttpStatus.NOT_FOUND));
    }

    @PostMapping("/add")
    public ResponseEntity<Flight> createFlight(@RequestBody Flight flight)
{
        Flight createdFlight = flightService.saveFlight(flight);
        return new ResponseEntity<>(createdFlight, HttpStatus.CREATED);
    }

    @DeleteMapping("/delete/{id}")
    public ResponseEntity<Void> deleteFlight(@PathVariable UUID id) {
        flightService.deleteFlight(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
    @PutMapping("/update")
    public ResponseEntity<Flight> updateFlight(@RequestBody Flight flight)
{
        Optional<Flight> existingFlight =
flightService.getFlightById(flight.getId());
        if (existingFlight.isPresent()) {
```

```
            Flight updatedFlight = existingFlight.get();
            updatedFlight.setSchedule(flight.getSchedule());
            updatedFlight.setDepartureCity(flight.getDepartureCity());
            updatedFlight.setDestinationCity(flight.getDestinationCity());
            updatedFlight.setType(flight.getType());
            updatedFlight.setPilot(flight.getPilot());
            updatedFlight.setCopilot(flight.getCopilot());
            updatedFlight.setStaffs(flight.getStaffs());

updatedFlight.setCateringCompanies(flight.getCateringCompanies());

            Flight savedFlight = flightService.saveFlight(updatedFlight);
            return new ResponseEntity<>(savedFlight, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

}
```

The Back-end side is characterized by the presence of a fairly solid security side by the presence of the JWT (JSON Web Token) represented by the security file and 2 filters, in order to give access to the urls according to different roles for the user :

Fichier sécurité

```
@Configuration @EnableWebSecurity @RequiredArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final UserDetailsService userDetailsService;
    private final BCryptPasswordEncoder bCryptPasswordEncoder;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {

auth.userDetailsService(userDetailsService).passwordEncoder(bCryptPasswordE
ncoder);
    }
    @Bean
    CorsConfigurationSource corsConfigurationSource() {
        UrlBasedCorsConfigurationSource source = new
                UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", new
CorsConfiguration().applyPermitDefaultValues());
        return source;
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.cors();
        http.sessionManagement().sessionCreationPolicy(STATELESS);

        http.authorizeRequests()
                .antMatchers("/login", "/user/token/refresh").permitAll()
                .antMatchers(HttpMethod.GET,
"/user/**").hasAuthority("ROLE_ADMIN")
                .antMatchers(HttpMethod.POST,
"/user/**").hasAuthority("ROLE_ADMIN")
                .antMatchers(HttpMethod.PUT,
```

```
"/user/**").hasAuthority("ROLE_ADMIN")
                .antMatchers(HttpMethod.DELETE,
"/user/**/**").hasAuthority("ROLE_ADMIN")

                .antMatchers(HttpMethod.GET,
"/pilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR", "ROLE_USER")
                .antMatchers(HttpMethod.POST,
"/pilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.PUT,
"/pilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.DELETE,
"/pilotes/**/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")

                .antMatchers(HttpMethod.GET,
"/copilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR",
"ROLE_USER")
                .antMatchers(HttpMethod.POST,
"/copilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.PUT,
"/copilotes/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.DELETE,
"/copilotes/**/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")

                .antMatchers(HttpMethod.GET,
"/staffs/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR", "ROLE_USER")
                .antMatchers(HttpMethod.POST,
"/staffs/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.PUT,
"/staffs/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.DELETE,
"/staffs/**/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")

                .antMatchers(HttpMethod.GET,
"/companies/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR",
"ROLE_USER")
                .antMatchers(HttpMethod.POST,
"/companies/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.PUT,
"/companies/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.DELETE,
"/companies/**/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")

                .antMatchers(HttpMethod.GET,
"/flights/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR", "ROLE_USER")
                .antMatchers(HttpMethod.POST,
"/flights/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.PUT,
"/flights/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")
                .antMatchers(HttpMethod.DELETE,
"/flights/**/**").hasAnyAuthority("ROLE_ADMIN", "ROLE_MODERATOR")

                .anyRequest().authenticated();

        http.addFilter(new
CustomAuthenticationFilter(authenticationManagerBean()));
        http.addFilterBefore(new CustomAuthorizationFilter(),
UsernamePasswordAuthenticationFilter.class);
    }
```

```
    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws
Exception {
        return super.authenticationManagerBean();
    }
}
```

Custom Authentification Filter

```
@Slf4j
public class CustomAuthenticationFilter extends
UsernamePasswordAuthenticationFilter {
    private final AuthenticationManager authenticationManager;
    public CustomAuthenticationFilter(AuthenticationManager
authenticationManager){
        this.authenticationManager= authenticationManager;
    }
    @Override
    public Authentication attemptAuthentication(HttpServletRequest request,
HttpServletResponse response) throws AuthenticationException {
        String username= request.getParameter("username");
        String password= request.getParameter("password");
        log.info("username : {}",username);
        log.info("password : {}",password);
        UsernamePasswordAuthenticationToken authenticationToken = new
UsernamePasswordAuthenticationToken(username,password);
        return authenticationManager.authenticate(authenticationToken);

    }

    @Override
    protected void successfulAuthentication(HttpServletRequest request,
HttpServletResponse response, FilterChain chain, Authentication
authentication) throws IOException, ServletException {
        User user = (User)authentication.getPrincipal();
        Algorithm algorithm = Algorithm.HMAC256("secret".getBytes());
        String access_token = JWT.create()
                .withSubject(user.getUsername())
                .withExpiresAt(new Date(System.currentTimeMillis() + 120 *
60 * 1000))
                .withIssuer(request.getRequestURL().toString())

.withClaim("roles",user.getAuthorities().stream().map(GrantedAuthority::get
Authority).collect(Collectors.toList()))
                .sign(algorithm);
        String refresh_token = JWT.create()
                .withSubject(user.getUsername())
                .withExpiresAt(new Date(System.currentTimeMillis() + 120 *
60 * 1000))
                .withIssuer(request.getRequestURL().toString())
                .sign(algorithm);
        /*response.setHeader("access_token",access_token);
        response.setHeader("refresh_token",refresh_token);*/
        Map<String , String> tokens = new HashMap<>();
        tokens.put("access_token",access_token);
        tokens.put("refresh_token",refresh_token);
        response.setContentType(MediaType.APPLICATION_JSON_VALUE);
        new ObjectMapper().writeValue(response.getOutputStream(),tokens);
    }}
```

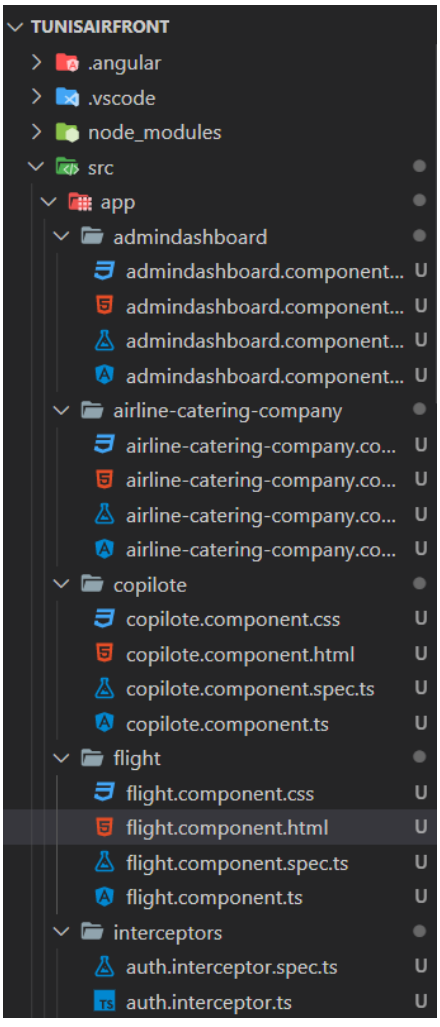
Custom Authorization Filter

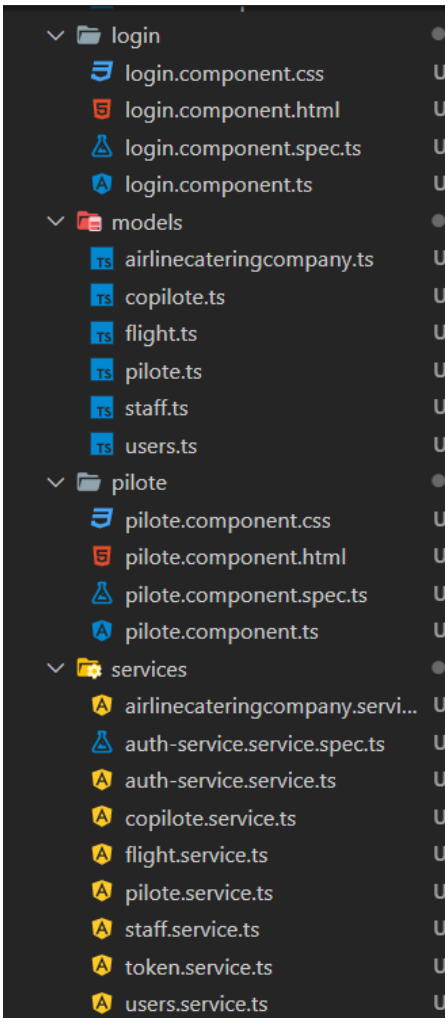
```
@Slf4j
```

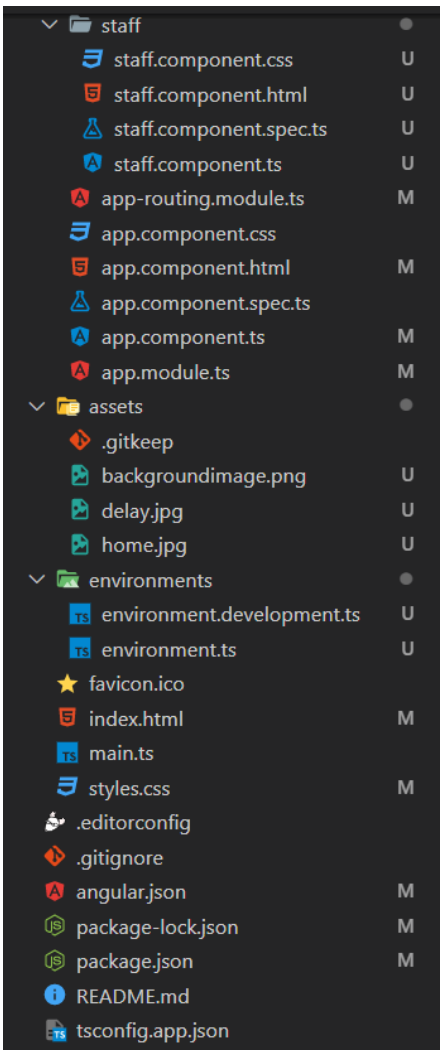
```
public class CustomAuthorizationFilter extends OncePerRequestFilter {
    @Override
    protected void doFilterInternal(HttpServletRequest request,
HttpServletResponse response, FilterChain filterChain) throws
ServletException, IOException {

if(request.getServletPath().equals("/login")||request.getServletPath().equa
ls("user/token/refresh")){
            filterChain.doFilter(request,response);
        }
        else {
            String authorizationHeader = request.getHeader(AUTHORIZATION);
            if(authorizationHeader != null &&
authorizationHeader.startsWith("Bearer ")){
                try{
                    String token = authorizationHeader.substring("Bearer
".length());
                    Algorithm algorithm =
Algorithm.HMAC256("secret".getBytes());
                    JWTVerifier verifier = JWT.require(algorithm).build();
                    DecodedJWT decodedJWT = verifier.verify(token);
                    String username = decodedJWT.getSubject();
                    String[] roles =
decodedJWT.getClaim("roles").asArray(String.class);
                    Collection<SimpleGrantedAuthority> authorities = new
ArrayList<>();
                    stream(roles).forEach(role->{
                        authorities.add(new SimpleGrantedAuthority(role));
                    });
                    UsernamePasswordAuthenticationToken authenticationToken
= new UsernamePasswordAuthenticationToken(username,null,authorities);

SecurityContextHolder.getContext().setAuthentication(authenticationToken);
                    filterChain.doFilter(request, response);
                }catch (Exception exception){
                log.error("Error Logging In : {}",exception.getMessage());
                response.setHeader("error", exception.getMessage());
                response.setStatus(FORBIDDEN.value());
                //response.sendError(FORBIDDEN.value());
                Map<String , String> error = new HashMap<>();
                error.put("error_message",exception.getMessage());
                response.setContentType(MediaType.APPLICATION_JSON_VALUE);
                new
ObjectMapper().writeValue(response.getOutputStream(),error);
                }

            }
            else {
                filterChain.doFilter(request, response);
            }
        }
    }
}
```

Front End

The architecture of the project in the front End is as follow:







Each class in the table and in the back is represented by an interface in the front:

```
export interface users {
    id: number;
    name: string;
    email: string;
    code: string;
    jobtitle: string;
    roles: Role[];
  }

  export interface Role {
    id: String|null;
    name: string;
  }
```

We have several components which represent the different components of the front end and connected with services for the processing and consumption of the Back end APIs, for example the connection part represented by the preceding user class, which once connects and using the JWT , retrieves the access Token and the stock in the cookies of the web page to be able to access the backend services, example:

Login.html: represents the authentication page, with its controller which is interested in processing the Html page and linked with the corresponding service:

```
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent{
  username: string = '';
  password: string = '';
  form!: FormGroup;
  constructor(
    private authService: AuthServiceService,
    private router: Router,
    private formBuilder: FormBuilder,
    private http:HttpClient,private tokenService: TokenService,
  ) {}
  ngOnInit(): void {
    this.form = this.formBuilder.group({
      username: '',
      password: ''
    });
  }




  login() {
    const formData = new URLSearchParams();
    formData.set('username', this.form.get('username')?.value);
    formData.set('password', this.form.get('password')?.value);

    this.http.post('http://localhost:8080/login', formData.toString(), {
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      withCredentials: true
    })
    .subscribe(
      (res: any) => {
        console.log(res.access_token);
        this.tokenService.setAccessToken(res.access_token);

        // Vérifier le rôle de l'utilisateur
        if (this.tokenService.hasRole('ROLE_ADMIN')) {
          this.router.navigate(['/admindashboard']);
        } else {
          this.router.navigate(['/pilote']);
        }

        console.log('Requête réussie !');
      },
```

```
      (error: any) => {
        console.log('Erreur lors de la requête :', error);
      }
    );
  }


  }
```

Service Login :

```
@Injectable({
  providedIn: 'root'
})
export class AuthServiceService {
  private baseUrl = 'http://localhost:8080/login'; // Update with your backend URL

  constructor(private http: HttpClient,private tokenService: TokenService) {}

  public logout(): void {
    // Supprimez les tokens et effectuez d'autres opérations nécessaires pour se déconnecter
    this.tokenService.removeAccessToken();
    // Autres opérations de déconnexion (si nécessaire)
  }
}
```

And the other components also have controllers with restrictions according to the roles of the users (example: user only has access to consultation, no management for location)

And the user roles are extracted directly from the access tokens which contains all the necessary information and these permissions.

Intelligence Artificielle

The last part of the development of our project will be devoted to the development of our artificial intelligence model and its training on the test base and on real data to predict flight delays according to several factors.

The first step to work on Google Colab will be the training of our model (according to the real data of our model the delays of tunisair flights are on average 48h):

+ Code + Texte

Reconnecter

```
InteractiveShell.ast_node_interactivity = "last_expr"
pd.options.display.max_columns = 50
%matplotlib inline
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/sample_data/Train.csv')
```

```
df.head()
```

	ID	DATOP	FLTID	DEPSTN	ARRSTN	STD	STA	STATUS	AC	target
0	train_id_0	2016-01-03	TU 0712	CMN	TUN	2016-01-03 10:30:00	2016-01-03 12.55.00	ATA	TU 32AIMN	260.0
1	train_id_1	2016-01-13	TU 0757	MXP	TUN	2016-01-13 15:05:00	2016-01-13 16.55.00	ATA	TU 31BIMO	20.0
2	train_id_2	2016-01-16	TU 0214	TUN	IST	2016-01-16 04:10:00	2016-01-16 06.45.00	ATA	TU 32AIMN	0.0
3	train_id_3	2016-01-17	TU 0480	DJE	NTE	2016-01-17 14:10:00	2016-01-17 17.00.00	ATA	TU 736IOK	0.0
4	train_id_4	2016-01-17	TU 0338	TUN	ALG	2016-01-17 14:30:00	2016-01-17 15.50.00	ATA	TU 320IMU	22.0

Création de la variable trajectoire : dep-arr

```
df['trajectory'] = df['DEPSTN'] + '-' + df['ARRSTN']
df_test['trajectory'] = df_test['DEPSTN'] + '-' + df_test['ARRSTN']
```

By training our model we have identified the most correlated variables with our results to know which are the most important factors in our prediction:

And after training, the last step will be to save our model and download it to our local pc so that we can use it and integrate it into our project:

```
[ ]  model.save_model('/content/sample_data/modele.cb')

[ ]  from google.colab import files
     files.download('/content/sample_data/modele.cb')
```

To run our model, we need a backend program to create the model's api that will be consumed by our Angular front end in order to predict the delays following the data we have, and the best backend to manage the models that has the Necessary library will be without Python support (Flask):

```python
import pandas as pd
from catboost import CatBoostRegressor  # (ou CatBoostClassifier pour la classification)
model = CatBoostRegressor()
model.load_model('C:/Users/MSI/OneDrive/Desktop/2 ING/Semestre 2/tp web/modele.cb')
new_data['DATOP'] = pd.to_datetime(new_data['DATOP'])
new_data['STD'] = pd.to_datetime(new_data['STD'])
new_data['STA'] = pd.to_datetime(new_data['STA'])
predictions = model.predict(new_data)
print(predictions)
from flask import Flask, request, jsonify
from flask_cors import CORS
app = Flask(__name__)
CORS(app)  # Active CORS pour toutes les routes de l'application

@app.route('/predictions', methods=['POST'])
def make_predictions():
    data = request.json  # Les données d'entrée doivent être envoyées au format JSON
    new_data = pd.DataFrame(data)
    new_data['DATOP'] = pd.to_datetime(new_data['DATOP'])
    new_data['STD'] = pd.to_datetime(new_data['STD'])
    new_data['STA'] = pd.to_datetime(new_data['STA'])
    predictions = model.predict(new_data)
    response = {'predictions': predictions.tolist()}
    return jsonify(response)

if __name__ == '__main__':
    app.run()
```

and finally it is the consumption of our API by a prediction function in the front end filled with the necessary data imported from our Angular user interface and the database managed by Spring Boot:

```typescript
public PredictFlight(formData: any) {
    const apiUrl = 'http://localhost:5000/predictions'; // L'URL de votre API Flask
    const schedule = new Date(formData.schedule).toISOString().substring(0, 10);
    // Récupérer la date et l'heure sélectionnées du champ schedule
    const selectedDate = new Date(formData.schedule);
    const year = selectedDate.getFullYear();
```

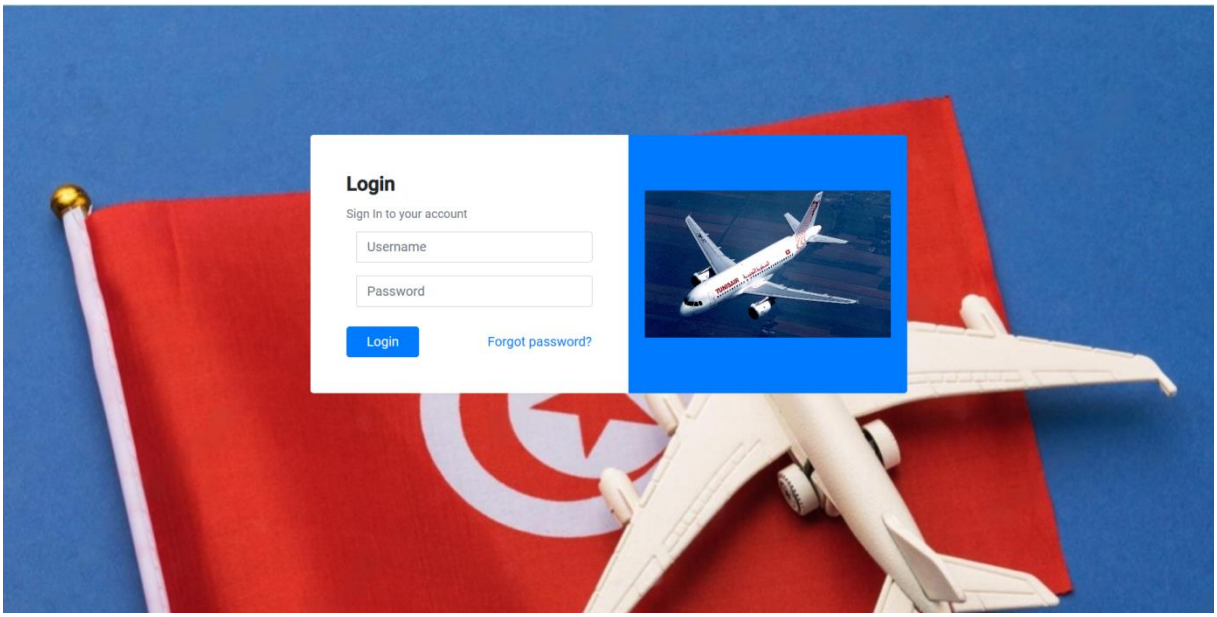
```
    const month = (selectedDate.getMonth() + 1).toString().padStart(2, '0'); // Les mois sont
indexés à partir de 0

      let season: number; // Variable pour stocker le numéro de la saison

      switch (month) {
        case "12":
        case "01":
        case "02":
          season = 4; // Hiver
          break;
        case "03":
        case "04":
        case "05":
          season = 1; // Printemps
          break;
        case "06":
        case "07":
        case "08":
          season = 2; // Été
          break;
        case "09":
        case "10":
        case "11":
          season = 3; // Automne
          break;
        default:
          season = -1; // Mois invalide
          break;
      }
    const day = selectedDate.getDate().toString().padStart(2, '0');
    const hours = selectedDate.getHours().toString().padStart(2, '0');
    const minutes = selectedDate.getMinutes().toString().padStart(2, '0');
    const seconds = selectedDate.getSeconds().toString().padStart(2, '0');
    // Format de la date et de l'heure au format "yyyy-MM-dd HH:mm:ss"
    const departuredateandtime = `${year}-${month}-${day} ${hours}:${minutes}:${seconds}`;
    const arrivalDate = (<HTMLInputElement>document.getElementById('arrdate')).value;

    const date = new Date(arrivalDate);
    const yeararr = date.getFullYear();
    const montharr = (date.getMonth() + 1).toString().padStart(2, '0');
    const dayarr = date.getDate().toString().padStart(2, '0');
    const hoursarr = date.getHours().toString().padStart(2, '0');
    const minutesarr = date.getMinutes().toString().padStart(2, '0');

    const arrivaldateandtime = `${yeararr}-${montharr}-${dayarr} ${hoursarr}:${minutesarr}:00`;
    const aircraftCodeElement = document.getElementById('Aircraftcode') as HTMLSelectElement;
    const selectedAircraftCode = aircraftCodeElement.value;
    const departureCity = formData.departureCity;
    const destinationCity = formData.destinationCity;
    const trajectory = departureCity + '-' + destinationCity;


    const departureDateforduration = new Date(departuredateandtime);
    const arrivalDateforduration = new Date(arrivaldateandtime);

    // Calculer la différence en millisecondes entre les deux dates
    const durationInMilliseconds = arrivalDateforduration.getTime() -
departureDateforduration.getTime();

    // Convertir la durée en heures, minutes et secondes
```

```
    const durationInSeconds = Math.floor(durationInMilliseconds / 1000);
    const durationInMinutes = Math.floor(durationInSeconds / 60);
    const durationInHours = Math.floor(durationInMinutes / 60);

    // Calculer les durées restantes après la conversion en heures
    const remainingMinutes = durationInMinutes % 60;
    const remainingSeconds = durationInSeconds % 60;
    let amOrPmdep: number; // Variable pour stocker la valeur 0 pour "AM" ou 1 pour "PM"

    if (departureDateforduration.getHours() < 12) {
      amOrPmdep = 0; // "AM"
    } else {
      amOrPmdep = 1; // "PM"
    }
    let amOrPmarr: number; // Variable pour stocker la valeur 0 pour "AM" ou 1 pour "PM"

    if (arrivalDateforduration.getHours() < 12) {
      amOrPmarr = 0; // "AM"
    } else {
      amOrPmarr = 1; // "PM"
    }
    const S_dep_hour = Math.sin((departureDateforduration.getHours() * 30 +
departureDateforduration.getMinutes() * 0.5) * (Math.PI / 180));
    const C_dep_hour = Math.cos((departureDateforduration.getHours() * 30 +
departureDateforduration.getMinutes() * 0.5) * (Math.PI / 180));
    const S_arr_hour = Math.sin((arrivalDateforduration.getHours() * 30 +
arrivalDateforduration.getMinutes() * 0.5) * (Math.PI / 180));
    const C_arr_hour = Math.cos((arrivalDateforduration.getHours() * 30 +
arrivalDateforduration.getMinutes() * 0.5) * (Math.PI / 180));
    const dayOfWeek = selectedDate.getDay();
    const weekOfYear = getISOWeek(selectedDate);
    const weekOfMonth = this.getWeekOfMonth(selectedDate);
    const dayOfYear = getDayOfYear(selectedDate);
    const requestData = {
      DATOP: [schedule],
      FLTID: ['TU 0850'],
      DEPSTN: [departureCity],
      ARRSTN: [destinationCity],
      STD: [departuredateandtime],
      STA: [arrivaldateandtime],
      STATUS: ['DEP'],
      AC: [selectedAircraftCode],
      trajectory: [trajectory],
      month: [month],
      day: [day],
      day_of_week: [dayOfWeek],
      year: [year],
      week_of_year: [weekOfYear],
      week_of_month: [weekOfMonth],
      season: [season],
      dep_hour: [hours],
      arr_hour: [hoursarr],
      dep_minute: [minutes],
      arr_minute: [minutesarr],
      flight_duration_sec: [remainingSeconds],
      flight_duration_hours: [durationInHours],
      flight_duration_minutes: [remainingMinutes],
      dep_hour_AM_PM: [amOrPmdep],
      arr_hour_AM_PM: [amOrPmarr],
      S_dep_hour: [S_dep_hour],
      C_dep_hour: [C_dep_hour],
```

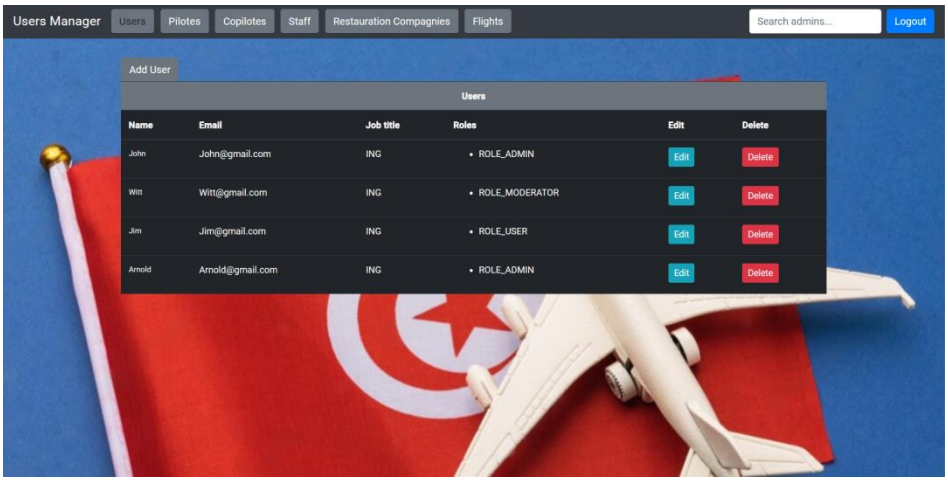
```
      S_arr_hour: [S_arr_hour],
      C_arr_hour: [C_arr_hour],
      day_of_year: [dayOfYear]
    };

    this.http.post(apiUrl, requestData).subscribe(
      (response: any) => {
        this.predictions = response.predictions; // Assignez les prédictions à la variable du
composant
        console.log(this.predictions); // Affiche les prédictions dans la console
      },
      (error) => {
        console.error('Une erreur s\'est produite lors de l\'appel à l\'API.', error);
      }
    );
  }
```

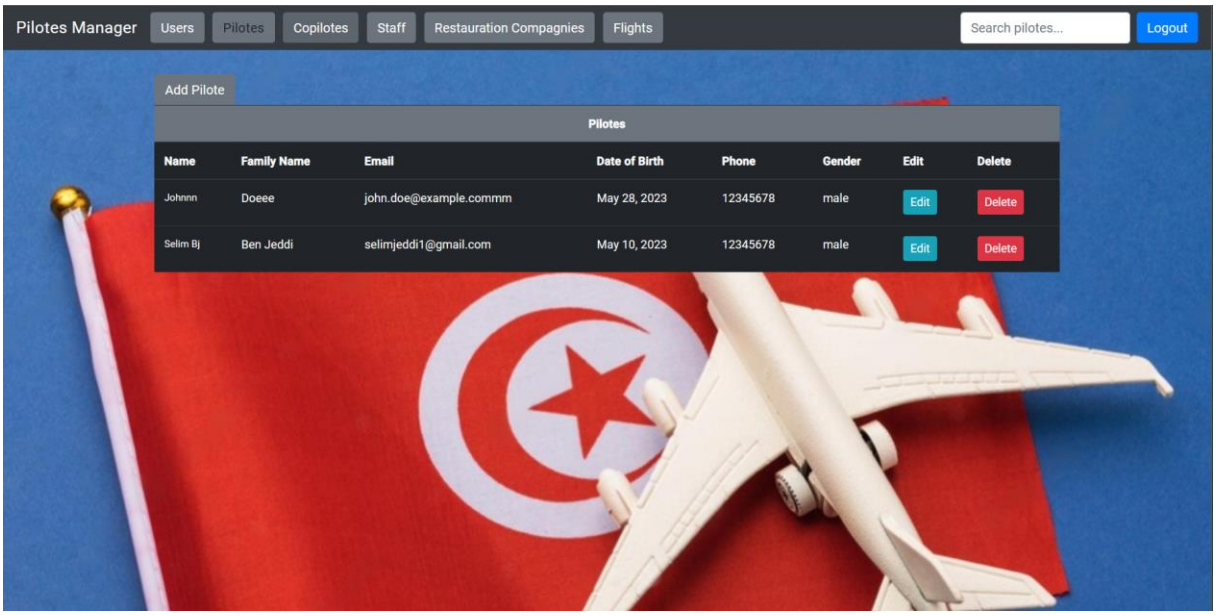
User Interfaces :

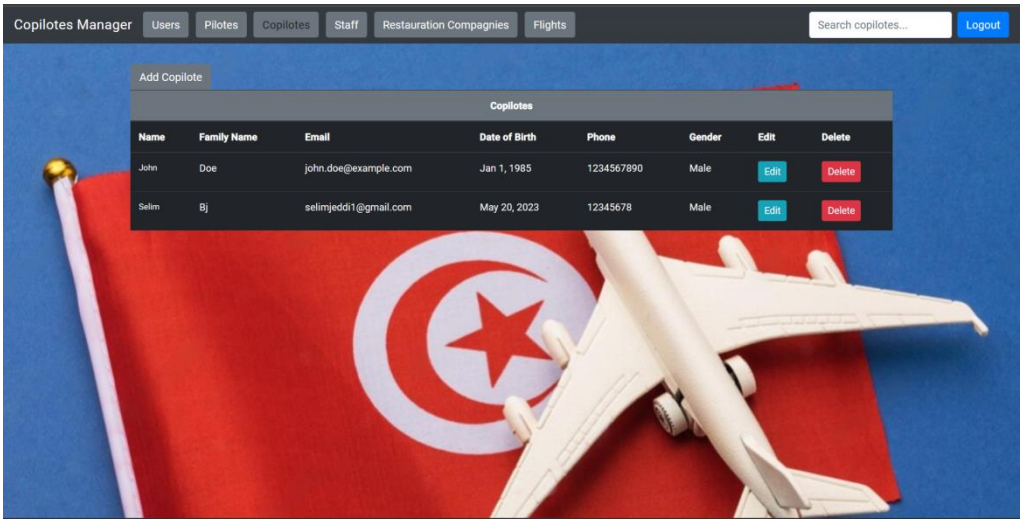
Login

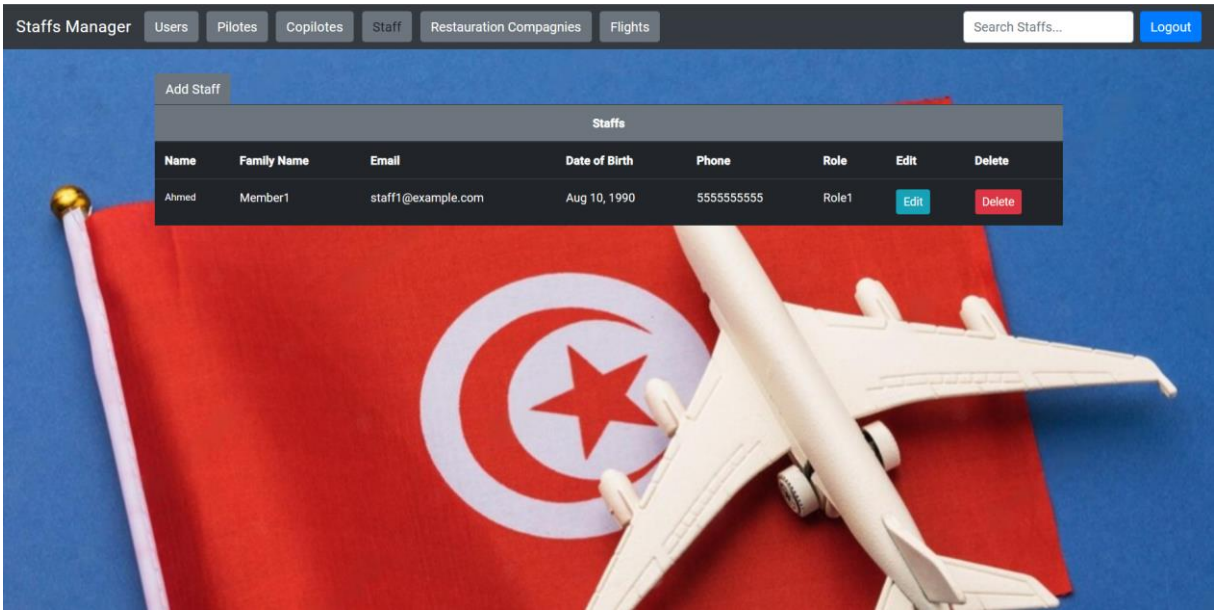


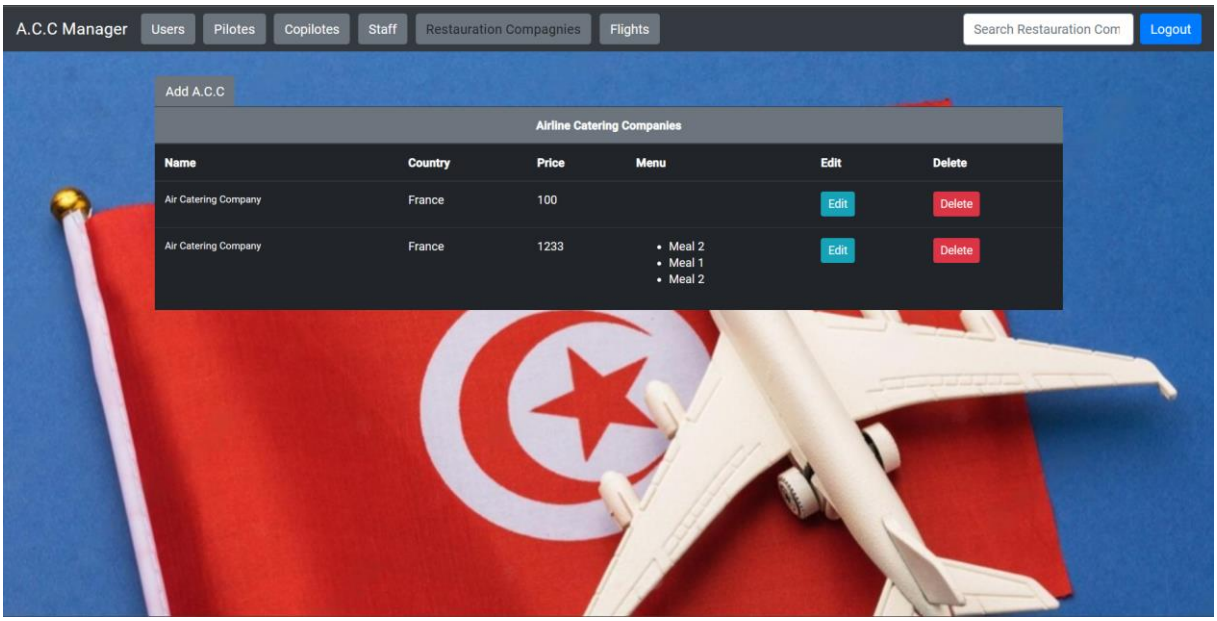
Admin Role

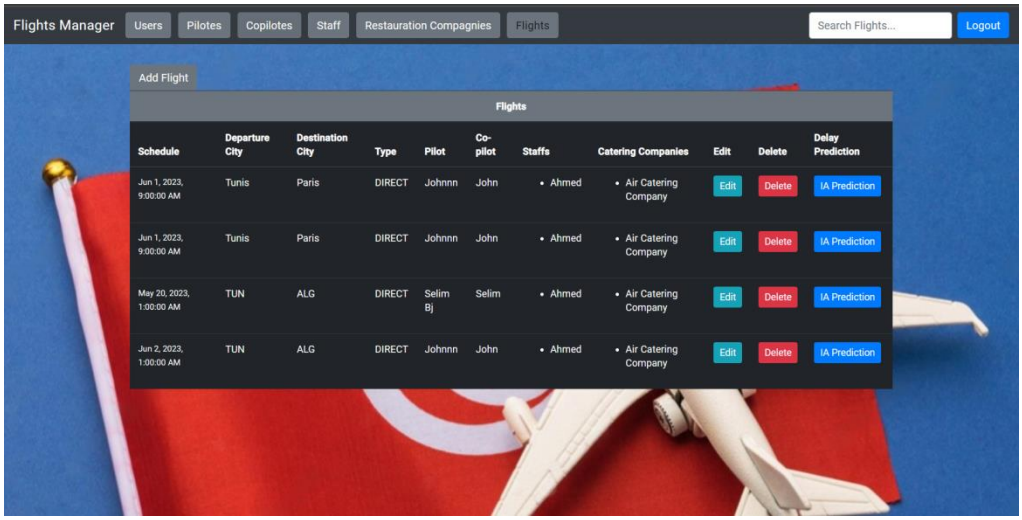


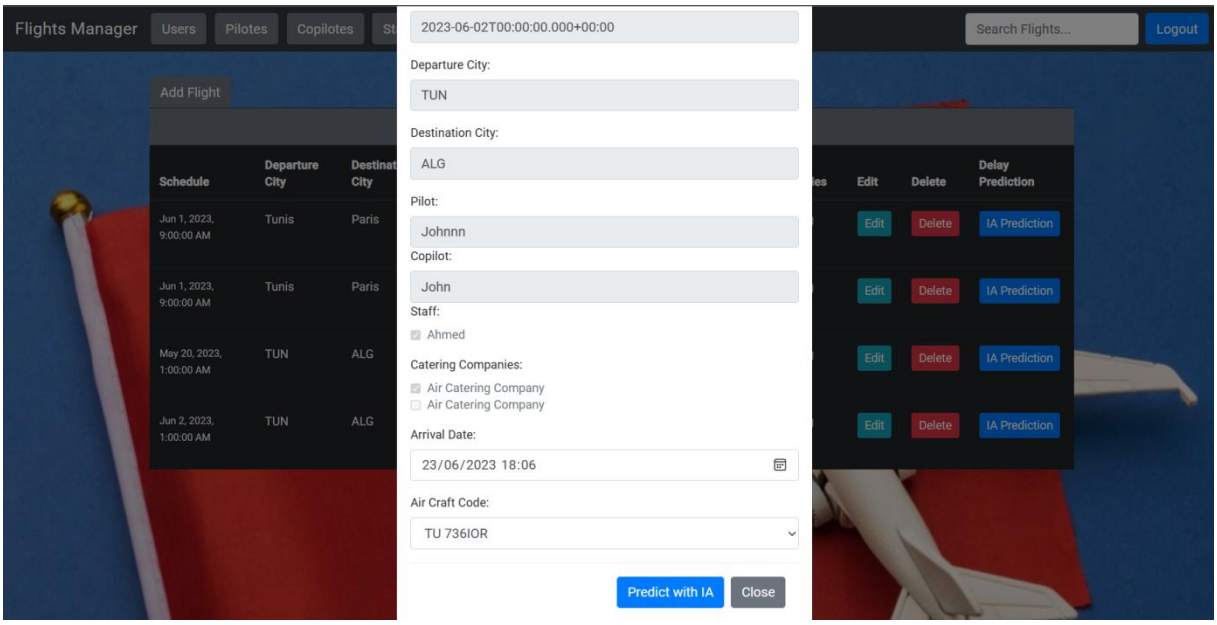

Pilotes Manager
Users
Pilotes
Copilotes
Staff
Restauration Compagnies
Flights
Search pilotes...
Logout
Add Pilote
Pilotes
Name
Family Name
Email
Date of Birth
Phone
Gender
Edit
Delete
Johnnn
Doeee
john.doe@example.commm
May 28, 2023
12345678
male
Edit
Delete
Selim Bj
Ben Jeddi
selimjeddi1@gmail.com
May 10, 2023
12345678
male
Edit
Delete

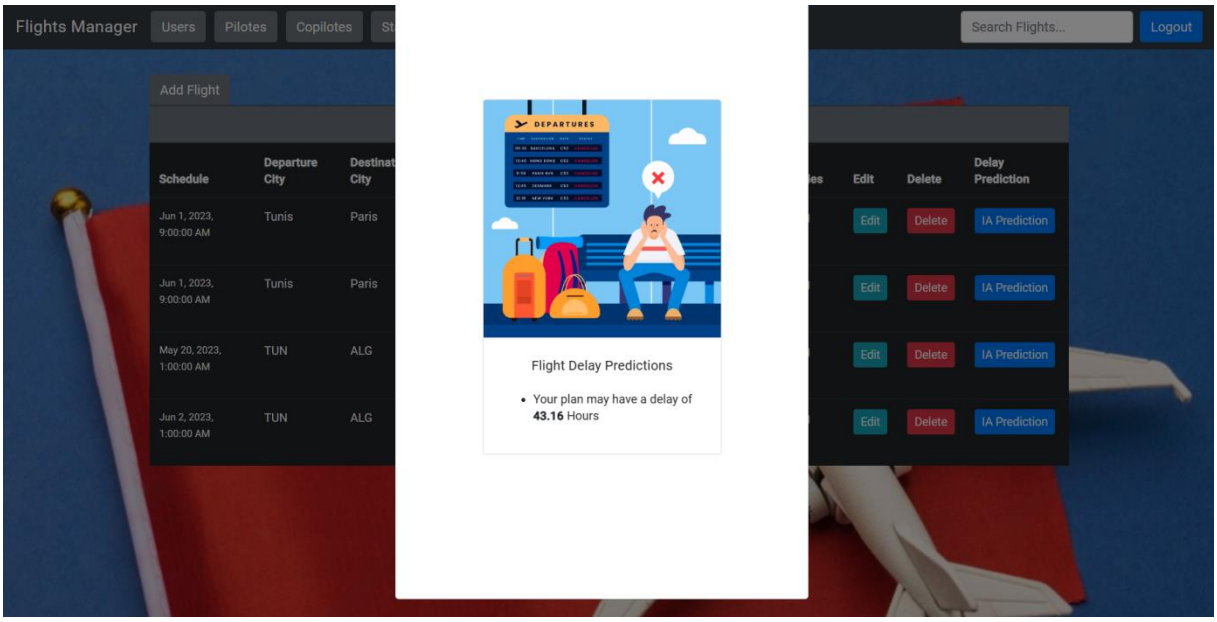

Copilotes Manager
Users
Pilotes
Copilotes
Staff
Restauration Compagnies
Flights
Search copilotes...
Logout
Add Copilote
Copilotes
Name
Family Name
Email
Date of Birth
Phone
Gender
Edit
Delete
John
Doe
john.doe@example.com
Jan 1, 1985
1234567890
Male
Edit
Delete
Selim
Bj
selimjeddi1@gmail.com
May 20, 2023
12345678
Male
Edit
Delete


Staffs Manager
Users
Pilotes
Copilotes
Staff
Restauration Compagnies
Flights
Search Staffs...
Logout
Add Staff
Staffs
Name
Family Name
Email
Date of Birth
Phone
Role
Edit
Delete
Ahmed
Member1
staff1@example.com
Aug 10, 1990
5555555555
Role1
Edit
Delete


A.C.C Manager
Users
Pilotes
Copilotes
Staff
Restauration Compagnies
Flights
Search Restauration Com
Logout
Add A.C.C
Airline Catering Companies
Name
Country
Price
Menu
Edit
Delete
Air Catering Company
France
100
Edit
Delete
Air Catering Company
France
1233
Meal 2
Meal 1
Meal 2
Edit
Delete

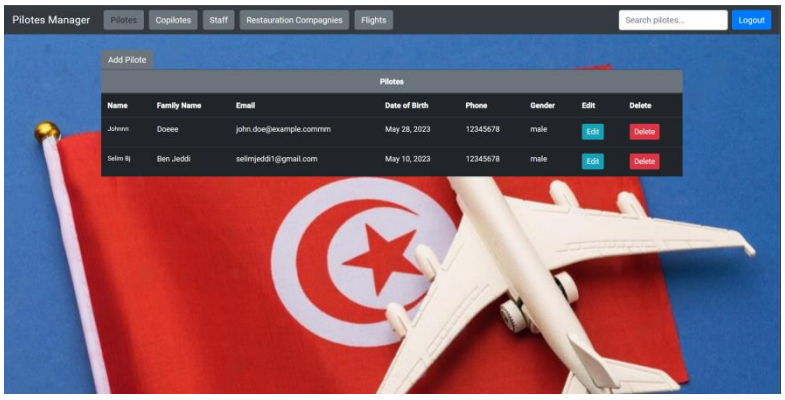

Flights Manager
Users
Pilotes
Copilotes
Staff
Restauration Compagnies
Flights
Search Flights...
Logout
Add Flight
Flights
Schedule
Departure City
Destination City
Type
Pilot
Co-pilot
Staffs
Catering Companies
Edit
Delete
Delay Prediction
Jun 1, 2023, 9:00:00 AM
Tunis
Paris
DIRECT
Johnnn
John
Ahmed
Air Catering Company
Edit
Delete
IA Prediction
Jun 1, 2023, 9:00:00 AM
Tunis
Paris
DIRECT
Johnnn
John
Ahmed
Air Catering Company
Edit
Delete
IA Prediction
May 20, 2023, 1:00:00 AM
TUN
ALG
DIRECT
Selim Bj
Selim
Ahmed
Air Catering Company
Edit
Delete
IA Prediction
Jun 2, 2023, 1:00:00 AM
TUN
ALG
DIRECT
Johnnn
John
Ahmed
Air Catering Company
Edit
Delete
IA Prediction





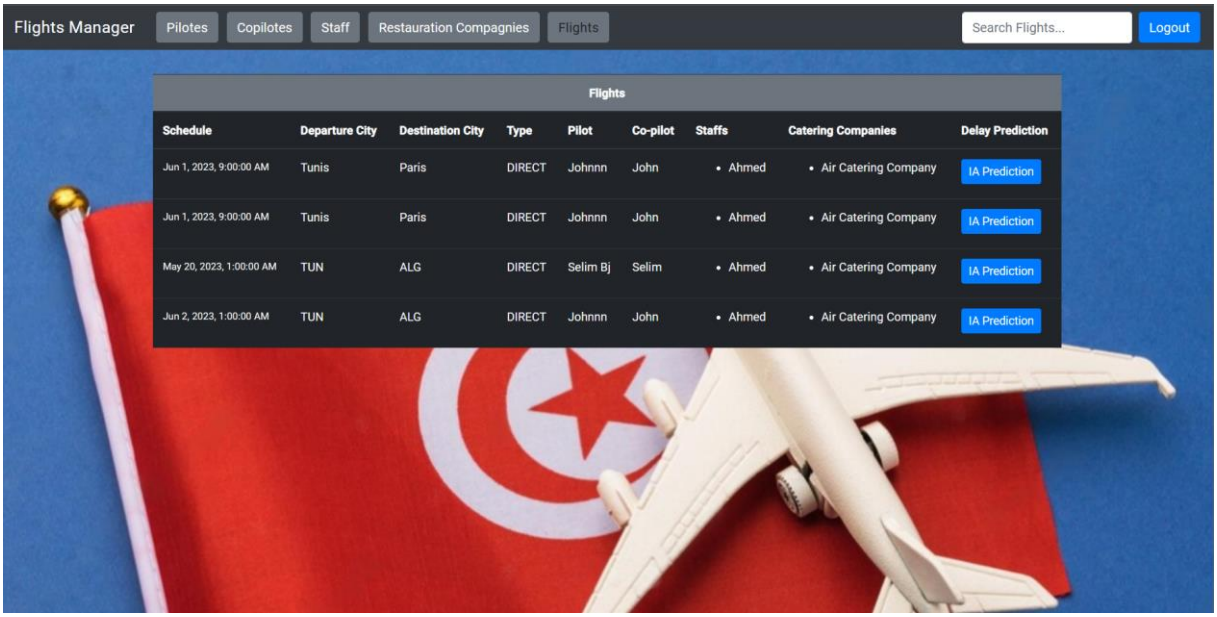
Moderator Role

The same interfaces as the Admin role, except that it does not have the user management space.



User Role

Has the right interfaces to consult and request an AI prediction, that's all.



Conclusion :

In conclusion, this report presents in depth the development project of a web application for the airline Tunisair. The project aims to improve the management of flights, aircrew and airline catering partners. The app incorporates an artificial intelligence module to predict flight delays, enabling better planning and a smoother travel experience for passengers.

Requirements analysis, architecture design, backend and frontend development, as well as the integration of the artificial intelligence model were presented in detail. The report also highlights the technologies used, such as Angular, Spring Boot, and Python.

This project demonstrates our ability to design and develop a complete web application that meets the specific requirements of Tunisair. We are proud of the results obtained and confident that this application will bring significant benefits, even in an academic context.

In summary, this project was an opportunity to acquire practical skills in web application development, artificial intelligence integration and project management. We hope that this work will serve as a solid foundation for future similar projects.