

SOMMAIRE

Introduction	2
Énoncé	3
Travail à faire	3
1. Schéma de l'architecture technique de l'application	Erreur ! Signet non défini.
2. Diagramme de classe	Erreur ! Signet non défini.
3. Couche DAO	Erreur ! Signet non défini.
a. Entité JPA	Erreur ! Signet non défini.
b. Interface JPA JpaRepository	Erreur ! Signet non défini.
c. Test de la couche DAO	Erreur ! Signet non défini.
4. Couche Web	Erreur ! Signet non défini.
d. Gérer les clients	Erreur ! Signet non défini.
e. Gérer les abonnements	Erreur ! Signet non défini.
5. Web service RESTful	Erreur ! Signet non défini.
6. Sécurité	Erreur ! Signet non défini.
Conclusion	20



Le présent rapport concerne le développement d'une application, basée sur Spring. Cette application permet de gérer les clients ainsi que leurs abonnements (type d'abonnement, solde, montant mensuel, etc.). Le développement de cette application a été réalisé en respectant une architecture technique basée sur un SGBD relationnel, Spring Data, JPA, Hibernate, Spring Security, et Angular.

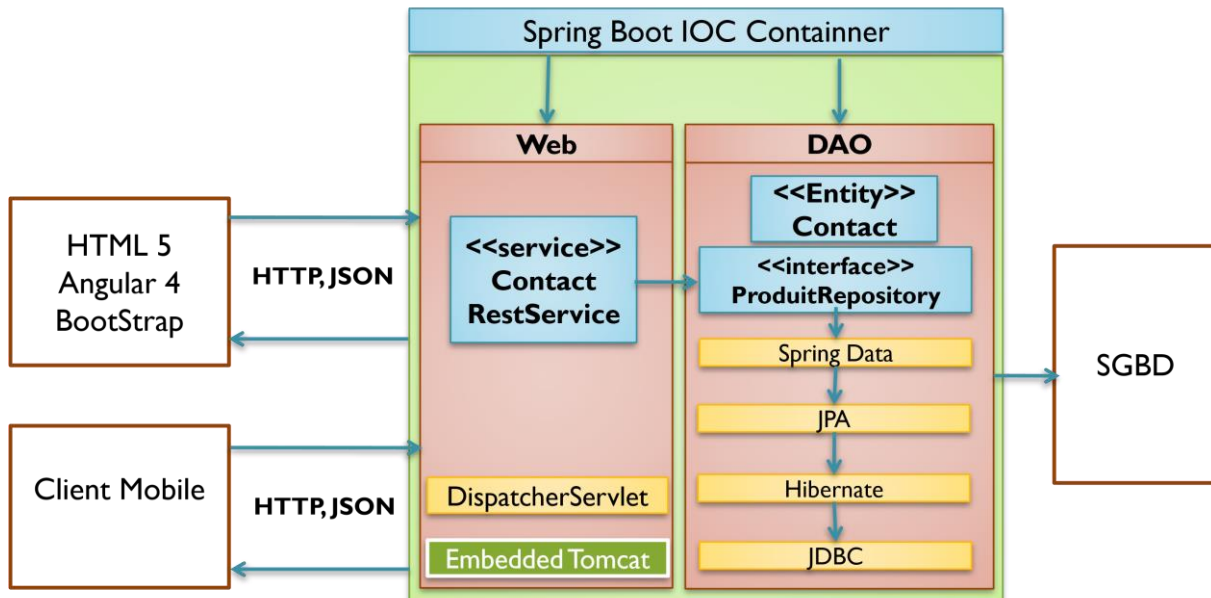
Dans ce rapport, nous allons présenter les différentes étapes de développement de l'application, en commençant par une description de l'architecture technique. Nous allons ensuite détailler les différentes fonctionnalités implémentées.

Enfin, nous allons conclure en présentant les résultats obtenus et les perspectives d'amélioration pour l'application.

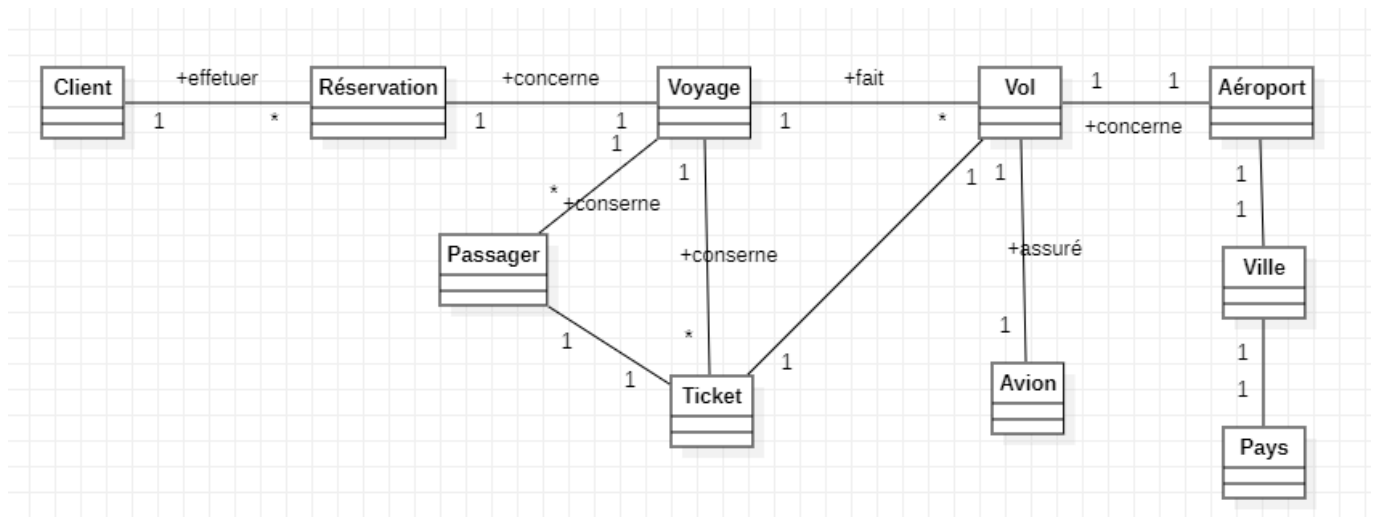
A. Conception :

1. Établir une architecture technique du projet

Micro Service



2. Établir un diagramme de classes qui montre les entités. On ne représentera que les attributs.



B. Implémentation

1. Créer un projet Spring boot avec les dépendances requises. Les identifiants du projet GroupId, ArtifactId et le package de base doivent contenir votre nom et prénom.

2. Couche DAO

1. Créer les entités JPA

```
package com.etoullali.entities;

import com.etoullali.enums.PositionGeographique;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Aeroport {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private PositionGeographique positionGeographique;
    @OneToOne
    private Vol vol;
    @OneToOne
    private Ville ville;
}
```

```
package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Avion {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private int nmrPlace;
    @OneToOne
    private Vol vol;
}
```

```
package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
```

```

import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String prenom;
    private String email;
    @OneToMany(mappedBy = "client")
    private List<Reservation> reservations;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Passager {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String email;
    @ManyToOne
    private Voyage voyage;
    @OneToOne
    private Ticket ticket;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Pays {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @OneToOne
    private Ville ville;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    private String siteReservation;
    @ManyToOne
    private Client client;
    @OneToOne
    private Voyage voyage;
}

```

```

package com.etoullali.entities;

import com.etoullali.enums.Type;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Ticket {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int numeroPlace;
    private Type prenom;
    @ManyToOne
    private Voyage voyage;
    @OneToOne
    private Vol vol;
    @OneToOne
    private Passager passager;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Entity
@Data
@AllArgsConstructor

```

```

@NoArgsConstructor
public class Ville {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @OneToOne
    private Aeroport aeroport;
    @OneToOne
    private Pays pays;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Vol {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private Date dateDepart;
    private Date dateArrivee;
    @ManyToOne
    private Voyage voyage;
    @OneToOne
    private Avion avion;
    @OneToOne
    private Aeroport aeroport;
    @OneToOne
    private Ticket ticket;
}

```

```

package com.etoullali.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.List;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Voyage {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @OneToOne
    private Reservation reservation;
    @OneToMany(mappedBy = "voyage")
    private List<Passager> passagers;
    @OneToMany(mappedBy = "voyage")
    private List<Ticket> tickets;
}

```



```

@OneToMany(mappedBy = "voyage")
private List<Vol> vols;
}

```

b. Créer les interfaces JPA Repository basées sur Spring Data

```

a.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AeroportRepository extends JpaRepository<Aeroport, Long> {
}

b.

c.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Avion;
import org.springframework.data.jpa.repository.JpaRepository;

public interface AvionRepository extends JpaRepository<Avion, Long> {
}

```

```

d.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Client;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ClientRepository extends JpaRepository<Client, Long> {
}

e.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Passager;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PassagerRepository extends JpaRepository<Passager, Long> {
}

```

```

f.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Pays;
import org.springframework.data.jpa.repository.JpaRepository;

public interface PaysRepository extends JpaRepository<Pays, Long> {
}

g.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Reservation;

```

```

import org.springframework.data.jpa.repository.JpaRepository;

public interface ReservationRepository extends JpaRepository<Reservation, Long> {
}

h.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Ticket;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TicketRepository extends JpaRepository<Ticket, Long> {
}

i.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Ville;
import org.springframework.data.jpa.repository.JpaRepository;

public interface VilleRepository extends JpaRepository<Ville, Long> {
}

j.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Vol;
import org.springframework.data.jpa.repository.JpaRepository;

public interface VolRepository extends JpaRepository<Vol, Long> {
}

k.      package com.etoullali.repositories;

import com.etoullali.entities.Aeroport;
import com.etoullali.entities.Voyage;
import org.springframework.data.jpa.repository.JpaRepository;

public interface VoyageRepository extends JpaRepository<Voyage, Long> {
}

```

c. Tester la couche DAO avec une application qui alimente la base de données avec quelques enregistrements de test.

Options

				id	date_arrivee	date_depart	nom	aeroport_id	avion_id	ticket_id	voyage_id
<input type="checkbox"/>	Éditer	Copier	Supprimer	1	2023-05-22 11:29:37	2023-05-22 11:29:37	vol 1	NULL	NULL	NULL	NULL
<input type="checkbox"/>	Éditer	Copier	Supprimer	2	2023-05-22 11:29:37	2023-05-22 11:29:37	vol 2	NULL	NULL	NULL	NULL
<input type="checkbox"/>	Éditer	Copier	Supprimer	3	2023-05-22 11:29:37	2023-05-22 11:29:37	vol 3	NULL	NULL	NULL	NULL

☐ Tout afficher
 Nombre de lignes : 25
 Filtrer les lignes: Chercher dans cette table
 Trier par clé : Aucun(e)

+ Options

				id	email	nom	prenom
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	1	ayoub@gmail.com	ETOULLALI	ayoub
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	2	hayat@gmail.com	ETOULLALI	hayat
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	3	samira@gmail.com	ETOULLALI	samira
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	4	mustapha@gmail.com	ETOULLALI	mustapha
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	5	karima@gmail.com	ETOULLALI	karima
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	6	radouan@gmail.com	ETOULLALI	radouan

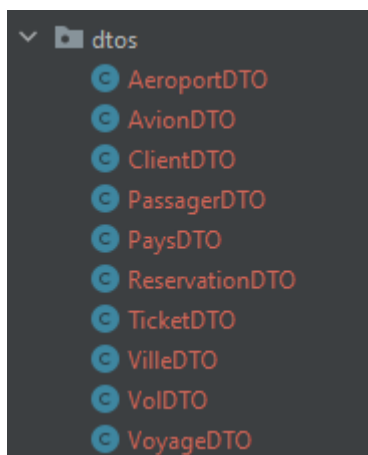
+ Options

				id	date_arrivee	date_depart	nom	aeroport_id	avion_id	ticket_id	voyage_id
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	1	2023-05-22 11:50:15	2023-05-22 11:50:15	vol 1	NULL	NULL	NULL	NULL
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	2	2023-05-22 11:50:16	2023-05-22 11:50:16	vol 2	NULL	NULL	NULL	NULL
<input type="checkbox"/>	✎ Éditer	📋 Copier	🗑 Supprimer	3	2023-05-22 11:50:16	2023-05-22 11:50:16	vol 3	NULL	NULL	NULL	NULL

Il est aussi possible de modifier la plupart

3. Couche Web REST API :

DTos



En créant les DTO et les mappeurs requis,

```
package com.etoullali.mappers;

import com.etoullali.dtos.*;
import com.etoullali.entities.*;
import org.springframework.beans.BeanUtils;
import org.springframework.stereotype.Service;

@Service
public class Mappers {

    public Aeroport fromAeroportDTO(AeroportDTO aeroportDTO) {
        Aeroport aeroport = new Aeroport();
        BeanUtils.copyProperties(aeroportDTO, aeroport);
        return aeroport;
    }
}
```

```

public AeroportDTO fromAeroport(Aeroport aeroport) {
    AeroportDTO aeroportDTO = new AeroportDTO();
    BeanUtils.copyProperties(aeroport, aeroportDTO);
    return aeroportDTO;
}

public Avion fromAvionDTO(AvionDTO avionDTO) {
    Avion avion = new Avion();
    BeanUtils.copyProperties(avionDTO, avion);
    return avion;
}

public AvionDTO fromAvion(Avion avion) {
    AvionDTO avionDTO = new AvionDTO();
    BeanUtils.copyProperties(avion, avionDTO);
    return avionDTO;
}

public Client fromClientDTO(ClientDTO clientDTO) {
    Client client = new Client();
    BeanUtils.copyProperties(clientDTO, client);
    return client;
}

public ClientDTO fromClient(Client client) {
    ClientDTO clientDTO = new ClientDTO();
    BeanUtils.copyProperties(client, clientDTO);
    return clientDTO;
}

public Passager fromPassagerDTO(PassagerDTO passagerDTO) {
    Passager passager = new Passager();
    BeanUtils.copyProperties(passagerDTO, passager);
    return passager;
}

public PassagerDTO fromPassager(Passager passager) {
    PassagerDTO passagerDTO = new PassagerDTO();
    BeanUtils.copyProperties(passager, passagerDTO);
    return passagerDTO;
}

public Pays fromPaysDTO(PaysDTO paysDTO) {
    Pays pays = new Pays();
    BeanUtils.copyProperties(paysDTO, pays);
    return pays;
}

public PaysDTO fromPays(Pays pays) {
    PaysDTO paysDTO = new PaysDTO();
    BeanUtils.copyProperties(pays, paysDTO);
    return paysDTO;
}

public Reservation fromReservationDTO(ReservationDTO reservationDTO) {
    Reservation reservation = new Reservation();
    BeanUtils.copyProperties(reservationDTO, reservation);
    return reservation;
}

public ReservationDTO fromReservation(Reservation reservation) {
    ReservationDTO reservationDTO = new ReservationDTO();
    BeanUtils.copyProperties(reservation, reservationDTO);
    return reservationDTO;
}

public Ticket fromTicketDTO(TicketDTO ticketDTO) {
    Ticket ticket = new Ticket();
    BeanUtils.copyProperties(ticketDTO, ticket);
    return ticket;
}

public TicketDTO fromTicket(Ticket ticket) {
    TicketDTO ticketDTO = new TicketDTO();

```

```

        BeanUtils.copyProperties(ticket, ticketDTO);
        return ticketDTO;
    }

    public Ville fromVilleDTO(VilleDTO villeDTO) {
        Ville ville = new Ville();
        BeanUtils.copyProperties(villeDTO, ville);
        return ville;
    }

    public VilleDTO fromVille(Ville ville) {
        VilleDTO villeDTO = new VilleDTO();
        BeanUtils.copyProperties(ville, villeDTO);
        return villeDTO;
    }

    public Vol fromVolDTO(VolDTO volDTO) {
        Vol vol = new Vol();
        BeanUtils.copyProperties(volDTO, vol);
        return vol;
    }

    public VolDTO fromVol(Vol vol) {
        VolDTO volDTO = new VolDTO();
        BeanUtils.copyProperties(vol, volDTO);
        return volDTO;
    }

    public Voyage fromVoyageDTO(VoyageDTO voyageDTO) {
        Voyage voyage = new Voyage();
        BeanUtils.copyProperties(voyageDTO, voyage);
        return voyage;
    }

    public VoyageDTO fromVoyage(Voyage voyage) {
        VoyageDTO voyageDTO = new VoyageDTO();
        BeanUtils.copyProperties(voyage, voyageDTO);
        return voyageDTO;
    }
}

```

- Créer le Web service RESTful qui permet de gérer les Vols

```

package com.etoullali.services;

import com.etoullali.dtos.*;
import com.etoullali.entities.Vol;

import java.util.List;

public interface VolService {
    VolDTO getVolById(Long id);
    List<VolDTO> getAllVols();

    void saveVol(VolDTO volDTO);
    void saveClient(ClientDTO client1);

    void saveVoyage(VoyageDTO voyageDTO);

    void savePassager(PassagerDTO passagerDTO);

    void saveTicket(TicketDTO ticketDTO);

    List<ClientDTO> getAllClients();

    ClientDTO getClientById(Long id);
}

```

```
List<VolDTO> searchVol(String keyword);
}
```

```
package com.etoullali.services;

import com.etoullali.dtos.*;
import com.etoullali.entities.*;
import com.etoullali.mappers.Mappers;
import com.etoullali.repositories.*;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.stream.Collectors;

@Service
@Transactional
@AllArgsConstructor
public class VolServiceImpl implements VolService {
    private VolRepository volRepository;
    private TicketRepository ticketRepository;
    private PassengerRepository passengerRepository;
    private VoyageRepository voyageRepository;
    private ClientRepository clientRepository;
    private Mappers mapper;
    @Override
    public VolDTO getVolById(Long id) {
        return mapper.fromVol(volRepository.findById(id).orElse(null));
    }

    @Override
    public List<VolDTO> getAllVols() {
        List<Vol> vols = volRepository.findAll();
        List<VolDTO> volDTOS = vols
            .stream()
            .map(vol -> mapper.fromVol(vol))
            .collect(Collectors.toList());
        return volDTOS;
    }

    @Override
    public void saveVol(VolDTO volDTO) {
        Vol vol=mapper.fromVolDTO(volDTO);
        volRepository.save(vol);
    }

    @Override
    public void saveClient(ClientDTO client1) {
        Client client=mapper.fromClientDTO(client1);
        clientRepository.save(client);
    }

    @Override
    public void saveVoyage(VoyageDTO voyageDTO) {
        Voyage voyage=mapper.fromVoyageDTO(voyageDTO);
        voyageRepository.save(voyage);
    }

    @Override
    public void savePassager(PassagerDTO passagerDTO) {
        Passager passager=mapper.fromPassagerDTO(passagerDTO);
        passagerRepository.save(passager);
    }
}
```

```

    }

    @Override
    public void saveTicket(TicketDTO ticketDTO) {

        Ticket ticket=mapper.fromTicketDTO(ticketDTO);
        ticketRepository.save(ticket);
    }

    @Override
    public List<ClientDTO> getAllClients() {
        List<Client> clients = clientRepository.findAll();
        List<ClientDTO> clientDTOS = clients
            .stream()
            .map(client -> mapper.fromClient(client))
            .collect(Collectors.toList());
        return clientDTOS;
    }

    @Override
    public ClientDTO getClientById(Long id) {
        return mapper.fromClient(clientRepository.findById(id).orElse(null));
    }

    @Override
    public List<VolDTO> searchVol(String keyword) {
        List<Vol> vols=volRepository.findByNomContains(keyword);
        List<VolDTO> volDTOS = vols.stream().map(vol ->
mapper.fromVol(vol)).collect(Collectors.toList());
        return volDTOS;
    }
}

```

```

package com.etoullali.web;

import com.etoullali.dtos.ClientDTO;
import com.etoullali.dtos.VolDTO;
import com.etoullali.services.VolService;
import lombok.AllArgsConstructor;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@AllArgsConstructor
@CrossOrigin("*")
public class VolController {
    private VolService volService;

    @GetMapping("/vols/all")
    public List<VolDTO> Vols() {
        return volService.getAllVols();
    }

    @GetMapping("/vols/{id}")
    public VolDTO getVol(@PathVariable(name = "id") Long id) {
        return volService.getVolById(id);
    }

    @GetMapping("/clients/all")
    public List<ClientDTO> Clients() {

```

```

        return volService.getAllClients();
    }

    @GetMapping("/vols/search")
    public List<VolDTO> searchCustomer(@RequestParam(defaultValue = "") String keyword) {
        return volService.searchVol(keyword); //"%"+keyword+"%"
    }
    @GetMapping("/clients/{id}")
    public ClientDTO getClient(@PathVariable(name = "id") Long id) {
        return volService.getClientById(id);
    }
}

```

- Créer le Web service RESTful qui permet de gérer les réservations

```

package com.etoullali.services;

import com.etoullali.dtos.*;

import java.util.List;

public interface ReservationService {
    ReservationDTO getReservationById(Long id);
    List<ReservationDTO> getAllReservations();
    void saveReservation(ReservationDTO reservationDTO);
}

```

```

package com.etoullali.services;

import com.etoullali.dtos.ReservationDTO;
import com.etoullali.dtos.VolDTO;
import com.etoullali.entities.Reservation;
import com.etoullali.entities.Vol;
import com.etoullali.mappers.Mappers;
import com.etoullali.repositories.ReservationRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;
import java.util.stream.Collectors;

@Service
@Transactional
@AllArgsConstructor
public class ReservationServiceImpl implements ReservationService {
    private Mappers mapper;
    private ReservationRepository reservationRepository;

    @Override
    public ReservationDTO getReservationById(Long id) {
        return
mapper.fromReservation(reservationRepository.findById(id).orElse(null));
    }

    @Override
    public List<ReservationDTO> getAllReservations() {

```



```

        List<Reservation> reservations = reservationRepository.findAll();
        List<ReservationDTO> reservationDTOS = reservations
            .stream()
            .map(reservation -> mapper.fromReservation(reservation))
            .collect(Collectors.toList());
        return reservationDTOS;
    }

    @Override
    public void saveReservation(ReservationDTO reservationDTO) {
        Reservation reservation=mapper.fromReservationDTO(reservationDTO);
        reservationRepository.save(reservation);
    }
}

```

```

package com.etoullali.web;

import com.etoullali.dtos.ClientDTO;
import com.etoullali.dtos.ReservationDTO;
import com.etoullali.dtos.VolDTO;
import com.etoullali.services.ReservationService;
import com.etoullali.services.VolService;
import lombok.AllArgsConstructor;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

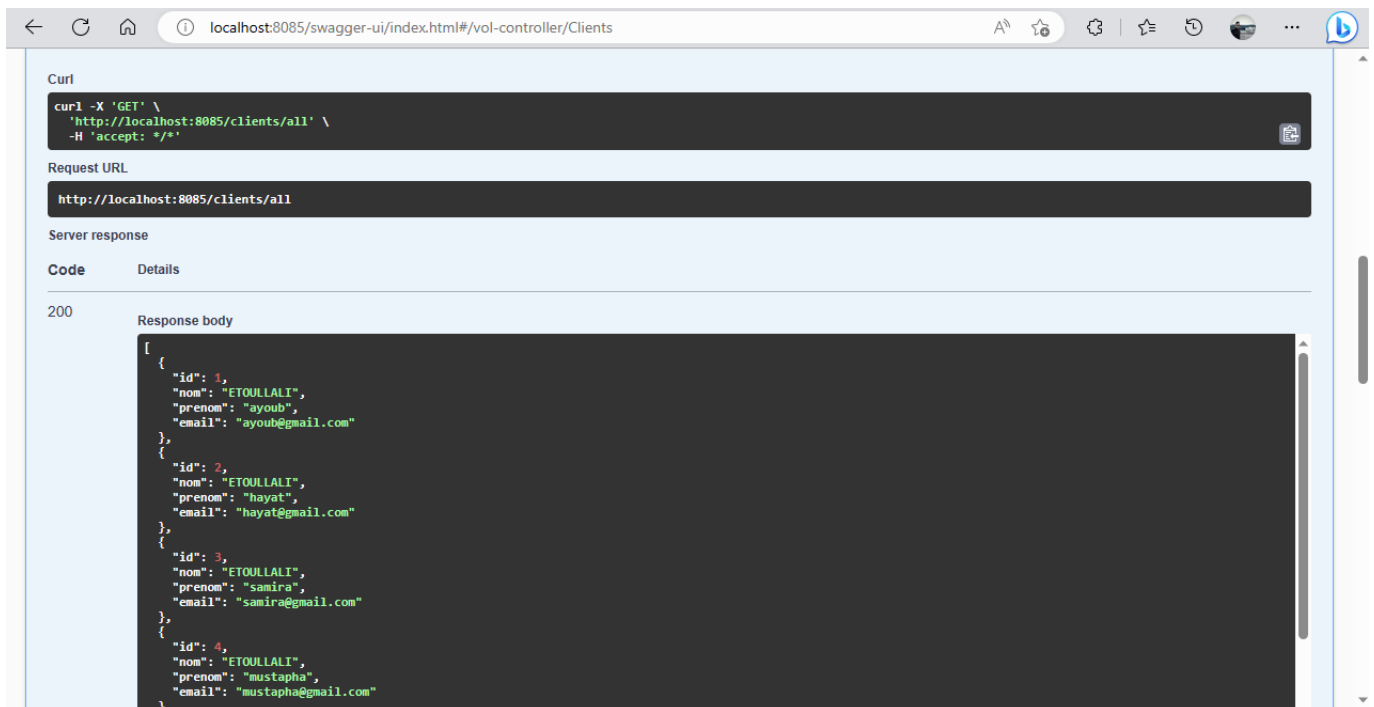
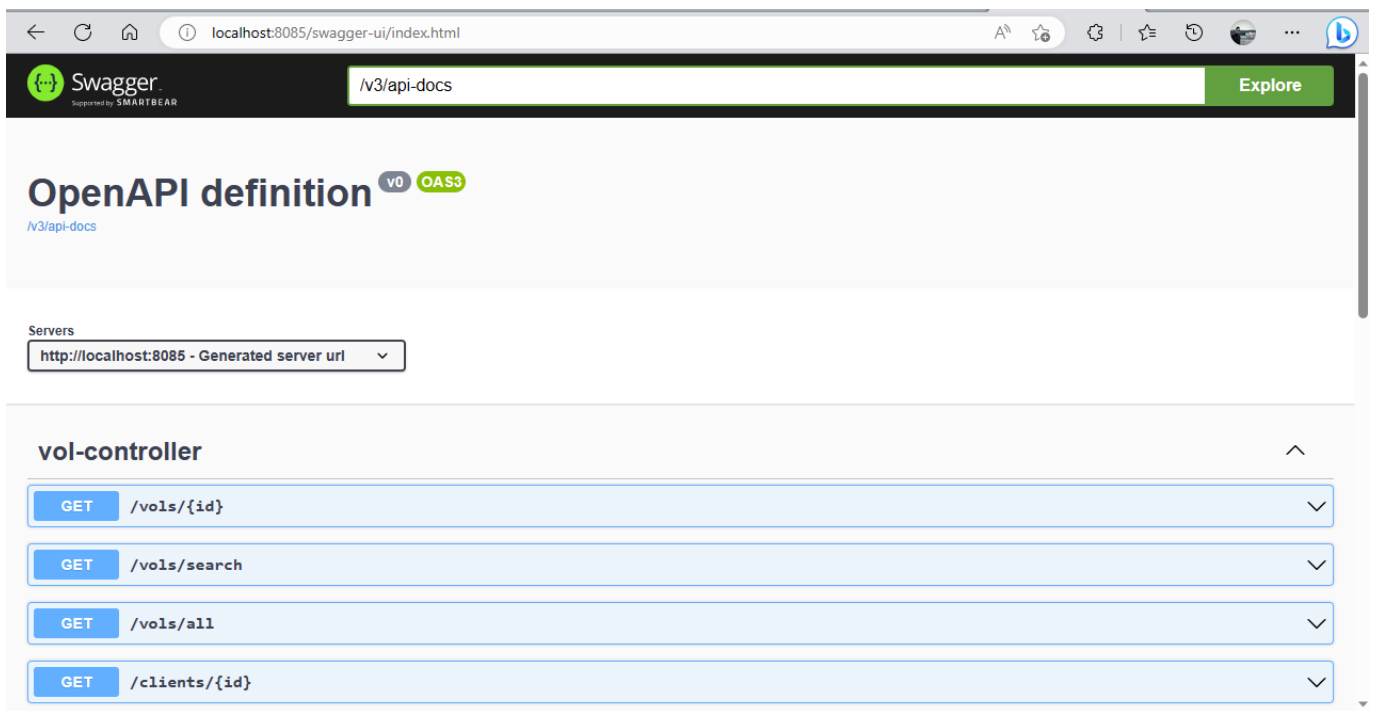
@RestController
@AllArgsConstructor
@CrossOrigin("*")
public class ReservationController {
    private ReservationService reservationService;

    @GetMapping("/reservations/all")
    public List<ReservationDTO> Reservations () {
        return reservationService.getAllReservations();
    }

    @GetMapping("/reservations/{id}")
    public ReservationDTO getReservation(@PathVariable(name = "id") Long id) {
        return reservationService.getReservationById(id);
    }
}

```

- Générer la documentation SWAGGER des API RESTful



Voir le document JSON dans le dossier examen

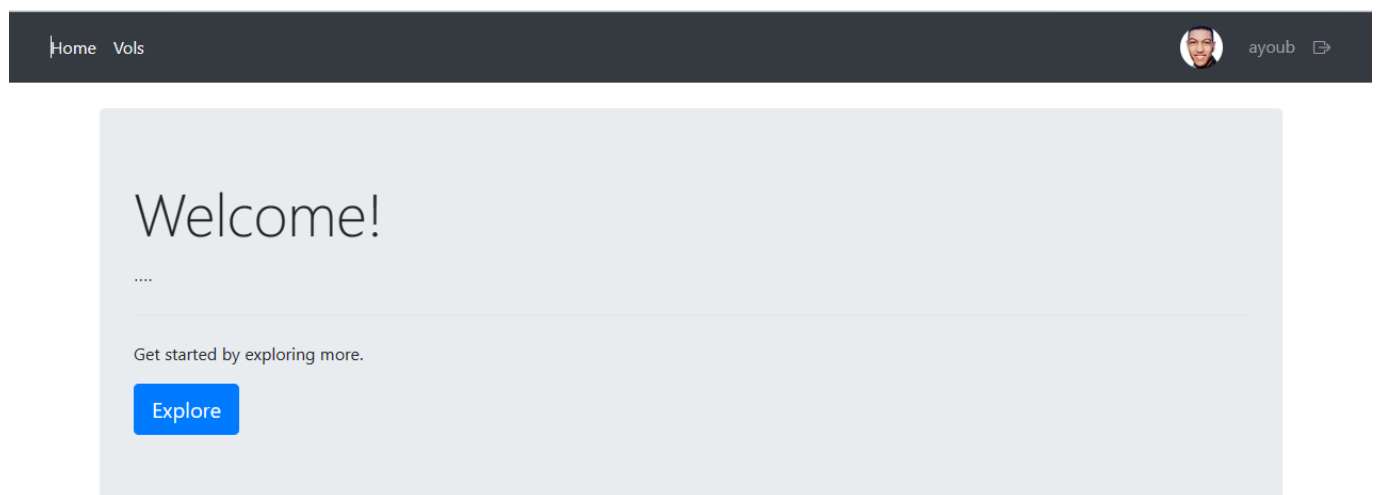
- Tester les Web service avec un client REST comme Postman

```
localhost:8085/reservations/all


1 [
2   {
3     "id": 1,
4     "date": null,
5     "siteReservation": "www.exemple1.com",
6     "clientDTO": null,
7     "voyageDTO": null
8   },
9   {
10    "id": 2,
11    "date": null,
12    "siteReservation": "www.exemple2.com",
13    "clientDTO": null,
14    "voyageDTO": null
15  }
16 ]
```

Tous les tests sont vérifier utilisons Swaquer

4. Proposer une application frontend en utilisant Angular Framework

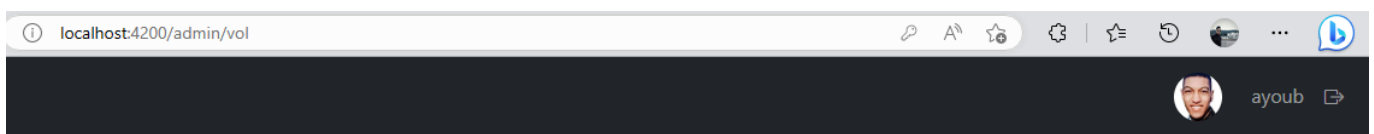


Customers			
<div>Keyword <input type="text"/></div>			
ID	Nom	date Depart	date Arrivee
1	vol 1	2023-05-22T10:50:15.000+00:00	2023-05-22T10:50:15.000+00:00
2	vol 2	2023-05-22T10:50:16.000+00:00	2023-05-22T10:50:16.000+00:00
3	vol 3	2023-05-22T10:50:16.000+00:00	2023-05-22T10:50:16.000+00:00

Keyword	2			
ID	Nom	date Depart	date Arrivee	
2	vol 2	2023-05-22T10:50:16.000+00:00	2023-05-22T10:50:16.000+00:00	

5. Sécuriser d'accès à cette application en se basant sur Spring security avec un système d'authentification des utilisateurs avec 3 types de rôles « ROLE CLIENT », «ROLE PASSAGER », « ROLE AEROPORT » et « ADMIN » en choisissant des autorisations appropriées à ses rôles

ADMIN



USER



Conclusion

En conclusion, le développement de l'application basée sur Spring a permis de mettre en pratique les différents concepts appris dans le cadre de notre formation en développement web.

Toutefois, il reste des perspectives d'amélioration pour cette application, telles que l'ajout de fonctionnalités supplémentaires, l'amélioration de l'interface utilisateur, la mise en place de tests automatisés, etc.

En somme, ce projet a été très enrichissant pour nous, car il nous a permis de consolider nos connaissances en développement web et en particulier en développement d'applications Spring. Nous espérons que ce rapport permettra de mieux comprendre les différentes étapes de développement de cette application et de ses fonctionnalités.