## Backend

1 ) Modification de la classe **UserController.java** pour ajouter le chemin **/export**

**ExportService**

public interface ExportService {

*/\*\*  
 \** ***@param*** *response  
 \** ***@throws*** *IOException  
 \** ***@throws*** *CsvDataTypeMismatchException  
 \** ***@throws*** *CsvRequiredFieldEmptyException  
 \*/*void exportUsersCSV(HttpServletResponse response) throws IOException, CsvDataTypeMismatchException, CsvRequiredFieldEmptyException;

**UserController.java :**

@Autowired  
private ExportService exportService;

*/\*\*  
 \* GET /export is a function that allows to export users in a csv file  
 \*  
 \** ***@param*** *response  
 \** ***@throws*** *IOException  
 \** ***@throws*** *CsvDataTypeMismatchException  
 \** ***@throws*** *CsvRequiredFieldEmptyException  
 \*/*@GetMapping(value="/export")  
public void exportUser(HttpServletResponse response) throws IOException, CsvDataTypeMismatchException, CsvRequiredFieldEmptyException {  
 exportService.exportUsersCSV(response);  
}

2) Création DTO **UserExportDTO.java** prendre exemple sur **StructureExportDTO.java** pour la structure des données de l’export (check)

public class UserExportDTO {  
  
 */\*\*  
 \* This annotation is used to define the columns of the csv file  
 \*/* @CsvBindByPosition(position = 0)  
 private String lastName;  
  
 @CsvBindByPosition(position = 1)  
 private String firstName;  
  
 @CsvBindByPosition(position = 2)  
 private String status;  
  
 @CsvBindByPosition(position = 3)  
 private String consentDate;  
  
 @CsvBindByPosition(position = 4)  
 private String checkPhone;  
  
 @CsvBindByPosition(position = 5)  
 private String checkMail;  
  
 @CsvBindByPosition(position = 6)  
 private String mail;  
  
 @CsvBindByPosition(position = 7)  
 private String telephoneNumber;  
  
 @CsvBindByPosition(position = 8)  
 private String siret;  
  
 @CsvBindByPosition(position = 9)  
 private String CompanyOrProjectName;  
  
 @CsvBindByPosition(position = 10)  
 private String inseeCode;  
  
 @CsvBindByPosition(position = 11)  
 private String nafCode;  
  
 //Getters and Setters  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String name) {  
 this.lastName = name;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status;  
 }  
  
 public String getConsentDate() {  
 return consentDate;  
 }  
  
 public void setConsentDate(String consentDate) {  
 this.consentDate = consentDate;  
 }  
  
 public String getCheckPhone() {  
 return checkPhone;  
 }  
  
 public void setCheckPhone(String checkPhone) {  
 this.checkPhone = checkPhone;  
 }  
  
 public String getCheckMail() {  
 return checkMail;  
 }  
  
 public void setCheckMail(String checkMail) {  
 this.checkMail = checkMail;  
 }  
  
 public String getMail() {  
 return mail;  
 }  
  
 public void setMail(String mail) {  
 this.mail = mail;  
 }  
  
 public String getTelephoneNumber() {  
 return telephoneNumber;  
 }  
  
 public void setTelephoneNumber(String telephoneNumber) {  
 this.telephoneNumber = telephoneNumber;  
 }  
  
 public String getSiret() {  
 return siret;  
 }  
  
 public void setSiret(String siret) {  
 this.siret = siret;  
 }  
  
 public String getCompanyOrProjectName() {  
 return CompanyOrProjectName;  
 }  
  
 public void setCompanyOrProjectName(String companyOrProjectName) {  
 CompanyOrProjectName = companyOrProjectName;  
 }  
  
 public String getInseeCode() {  
 return inseeCode;  
 }  
  
 public void setInseeCode(String inseeCode) {  
 this.inseeCode = inseeCode;  
 }  
  
 public String getNafCode() {  
 return nafCode;  
 }  
  
 public void setNafCode(String nafCode) {  
 this.nafCode = nafCode;  
 }  
  
}

3 ) Création classe **ExportUser.java** prendre exemple sur **ExportStructure.java** (permet de donner le nom des en-têtes des colonnes de l’export)

*/\*\*  
 \* set the name of the columns in the csv file  
 \*  
 \** ***@param*** <*UserExportDTO*>  
 *\*/*public class ExportUser<UserExportDTO> extends ColumnPositionMappingStrategy<UserExportDTO> {  
 private static final String[] *HEADER* = new String[]{"Nom usager", "Prénom", "Statut", "Date de consentement",  
 "Coche téléphone", "Coche mail", "Mail", "Tel", "SIRET", "Raison sociale/Nom projet",  
 "Code insee localisation", "Code naf"};  
  
 @Override  
 public String[] generateHeader(UserExportDTO dto) throws CsvRequiredFieldEmptyException {  
 super.generateHeader(dto);  
 return *HEADER*;  
 }  
}

4) Modification classe **AppAccessControl.java** pour prendre en compte le nouveau chemin

C’est bien pour ADMINAGENT ?

public class AppAccessControl {

private static AccessControl setAccessControl() {

AccessControl ac = new AccessControl();

// UserController  
ac.forRequest("/utilisateurs", AccessControl.*AUTHENTICATED*);  
ac.forMethod(HttpMethod.*DELETE*, "/utilisateurs/{id}", AccessControl.*ROLE\_ADMINAGENT*);  
ac.forMethod(HttpMethod.*GET*, "/keycloakid/{keycloakid}", AccessControl.*AUTHENTICATED*);  
ac.forMethod(HttpMethod.*GET*, "/utilisateurs/export", AccessControl.*ROLE\_ADMINAGENT*);

5) Modification classe **ExportServiceImpl.java** pour pouvoir mapper les différences entre l’entité et le DTO

ExportService

public interface ExportService {

void exportUsersCSV(HttpServletResponse response) throws IOException, CsvDataTypeMismatchException, CsvRequiredFieldEmptyException;

**ExportServiceImpl**

@Service  
public class ExportServiceImpl implements ExportService {

private ModelMapper modelMapper;

private UserService userService;

@Override  
public void exportUsersCSV(HttpServletResponse response) throws IOException, CsvDataTypeMismatchException, CsvRequiredFieldEmptyException {  
 Type listType = new TypeToken<List<UserExportDTO>>() {}.getType();  
 ArrayList<User> users = (ArrayList<User>) userService.findAll();  
 ArrayList<UserExportDTO> dtos = modelMapper.map(users, listType);  
  
 String filename = "export.csv";  
  
 response.setContentType("text/csv");  
 response.setCharacterEncoding("CP1252");  
 response.setHeader(HttpHeaders.*CONTENT\_DISPOSITION*,  
 "attachment; filename=\"" + filename + "\"");  
  
  
 ExportUser<UserExportDTO> mappingStrategy = new ExportUser<>();  
 mappingStrategy.setType(UserExportDTO.class);  
  
 //create a csv writer  
  
 StatefulBeanToCsv<UserExportDTO> writer = new StatefulBeanToCsvBuilder<UserExportDTO>(response.getWriter())  
 .withMappingStrategy(mappingStrategy)  
 .withQuotechar(CSVWriter.*DEFAULT\_QUOTE\_CHARACTER*)  
 .withEscapechar('"')  
 .withSeparator(';')  
 .withOrderedResults(false)  
 .build();  
  
 writer.write(dtos);  
}

UserService

public interface UserService {

Collection<User> findAll();

**ATTENTION : La construction de se mapper de DTO est la partie incertaine de développement**

+ Test unitaire (Récupération du DTO)

## Frontend

Modification du fichier **card-menu.composent.html** pour ajouter un nouveau type de carte avec un bouton extraction qui appel une fonction d’export (+ SCSS ?)

Modification du fichier **home.connected.component.html** pour ajouter la carte

Modification du fichier **home.connected.component.ts** pour ajouter la fonction d’export (prendre exemple de la fonction **exportStructures()** dans le fichier **structure.component.ts**)

Création d’un **userservice.ts** pour appeler le **path** envoyé par le back

Modification du **fr.json** pour mettre les information de la nouvelle carte (label, icone, etc…).

Modification du fichier **index.ts** pour mettre les informations de la nouvelle carte (label, icone, etc…).

Préconisation DEV/TU : 3 + 1 = 4J