Projet de développement d’application Desktop

Rudolph Attiso & Youssef Lasri
Apprentis Concepteur et Développeur d’applications en herbe

Cahier des charges : Javacard

1. Introduction

2. Analyse Fonctionnelle..
   - Création du “persona”.
   - Construction d‘un Sprint pour la création du backlog avec le user stories.

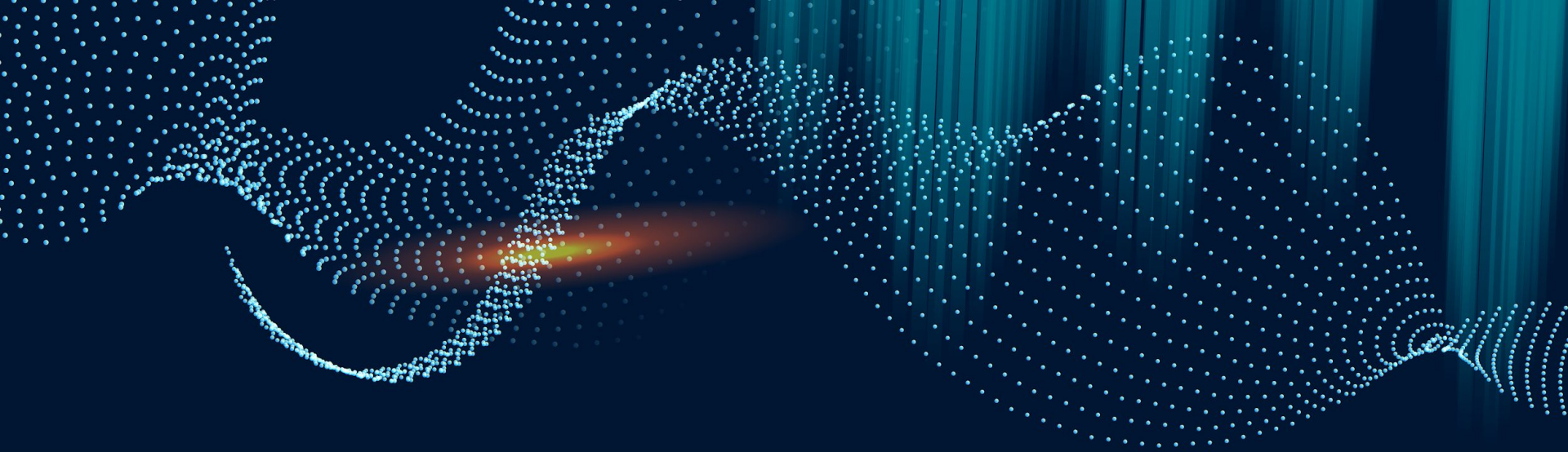
3. Maquettage de l’application.
   - Création du Zoning et du Wireframe.
   - Validation de l’interface avec le product owner.

4. Conception de l’application.
   - Réflexion architecturale.
   - Conception de diagramme.

5. Début des développements.

6. Test fonctionnels au cours des développements.

7. Conclusion
   - Livraison de l’applications.



1 Introduction

- Methodes de developpement application Desktop.

L’objectif de ce projet est de développer une application “desktop” Java permettant de gérer une liste de contacts et d’exporter les informations

Choix technologiques :

• l’application doit être codée en Java (version 17 ou ultérieure) • l’interface graphique devra être construite avec JavaFX

• les vues de l’application devront être codées en FXML

• pas de contrainte sur les bibliothèques utilisables

Modalités de travail :

• date de début du projet : 26/07/2024

• présentation du projet : 07/08/2024

• équipes de 2

• gestion du projet inspiré par la méthode Scrum • utilisation d’un dépôt Git partagé

Livrables attendus :

• code source de l’application

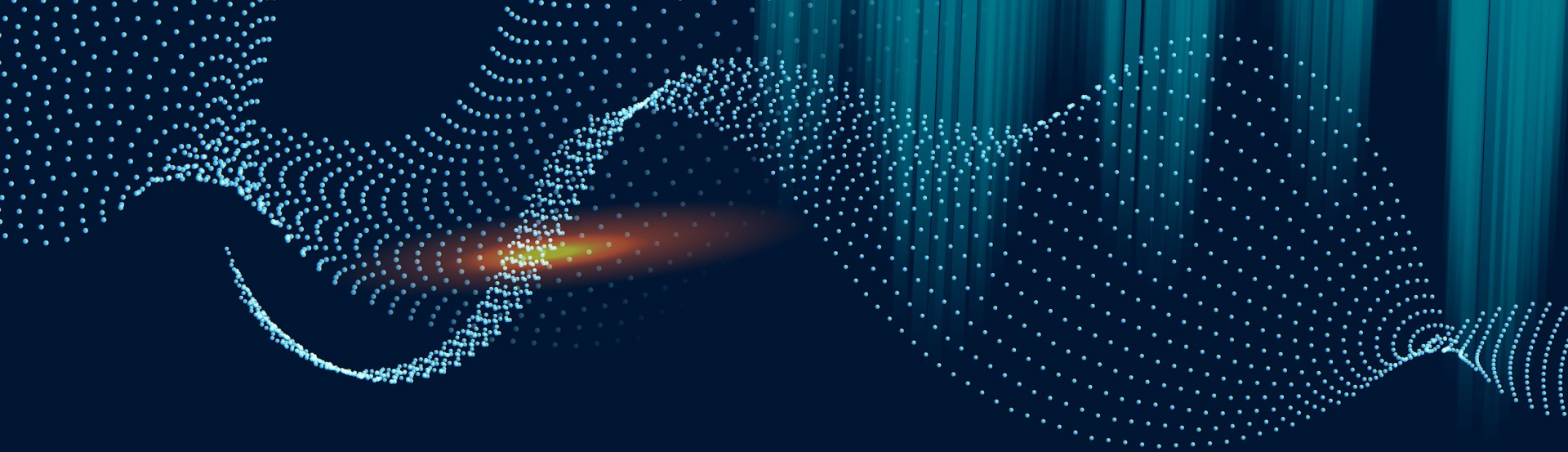
• diagrammes UML éventuels

• “zoning” et “wireframe” de l’interface graphique

Les fonctionnalités principales attendues pendant le Sprint définie à l'aide de l'indicateur MoSCoW :

Fonctionnalité	MoSCoW
Interface graphique ergonomique proposant une gestion des contacts (CRUD sur les contacts)	M
Sauvegarde des informations de contact pour les recharger lorsque le logiciel se lance	M
Export d'un ou plusieurs contacts en JSON	M
Export d'un ou plusieurs contacts en vCard	M
Rechercher dans la liste des contact	S
Générer un QRCode contenant les informations de la vCard et l'afficher à l'écran	C
Export d'un ou plusieurs contacts en CSV	C

Dans la suite de ce document, vous trouverez des informations détaillées sur ces différentes fonctionnalités.



2 Analyse fonctionnelle

- Création du persona
- Construction d’un backlog avec le user stories
- Mise en place du Dépôt Git partagé

Création du persona

Julie Martin

Intitulé de poste

Chef de Projet

Âge

Entre 25 et 34 ans

Niveau d'études

Licence ou diplôme équivale

Réseaux sociaux

Secteur d'activité

Technologie

Moyen de communication préféré

- Téléphone
- E-mail
- Réseaux sociaux

Outils nécessaires au quotidien

- Système de gestion de contenu
- Outil de gestion de projet
- Applications de stockage et de partage de dossiers en ligne

Responsabilités

Saisissez un texte

Indicateurs de performance

Données sur les nouveaux visiteurs, Vérification de l'identité des utilisateurs accédant à une application.

Supérieur hiérarchique

Directeur de Projet

Taille de l'entreprise

11 à 50 salariés

Objectifs

Pouvoir utiliser application desktop permettant de gérer les conatcs et de les exporter

Sources d'information

Saisissez un texte

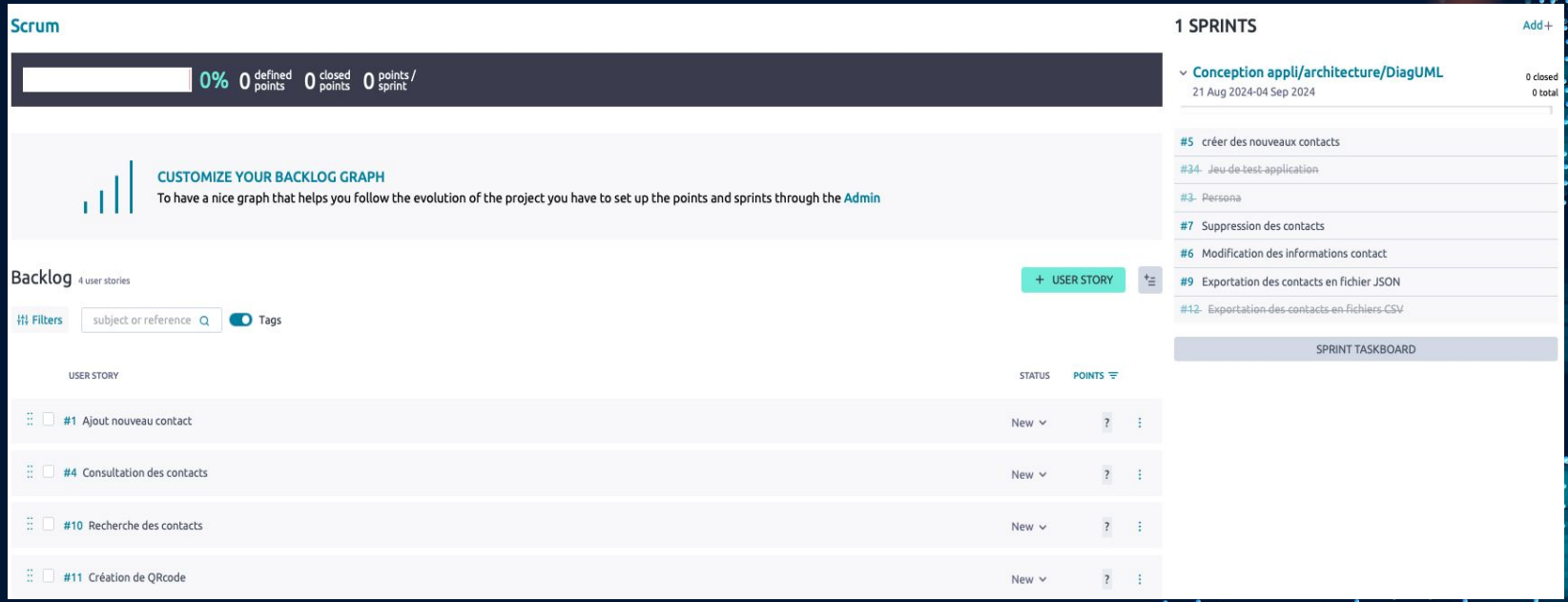
Principaux défis

- Ressources
- Gestion des changements
- Communication interne
- Gestion des projets et organisation
- Développement professionnel
- Résolution des problèmes et prise de décisions

Construction d’un Sprint pour la création du Backlog et de User Stories

Lien: https://tree.taiga.io/project/rudolphattisso-contact_management/backlog

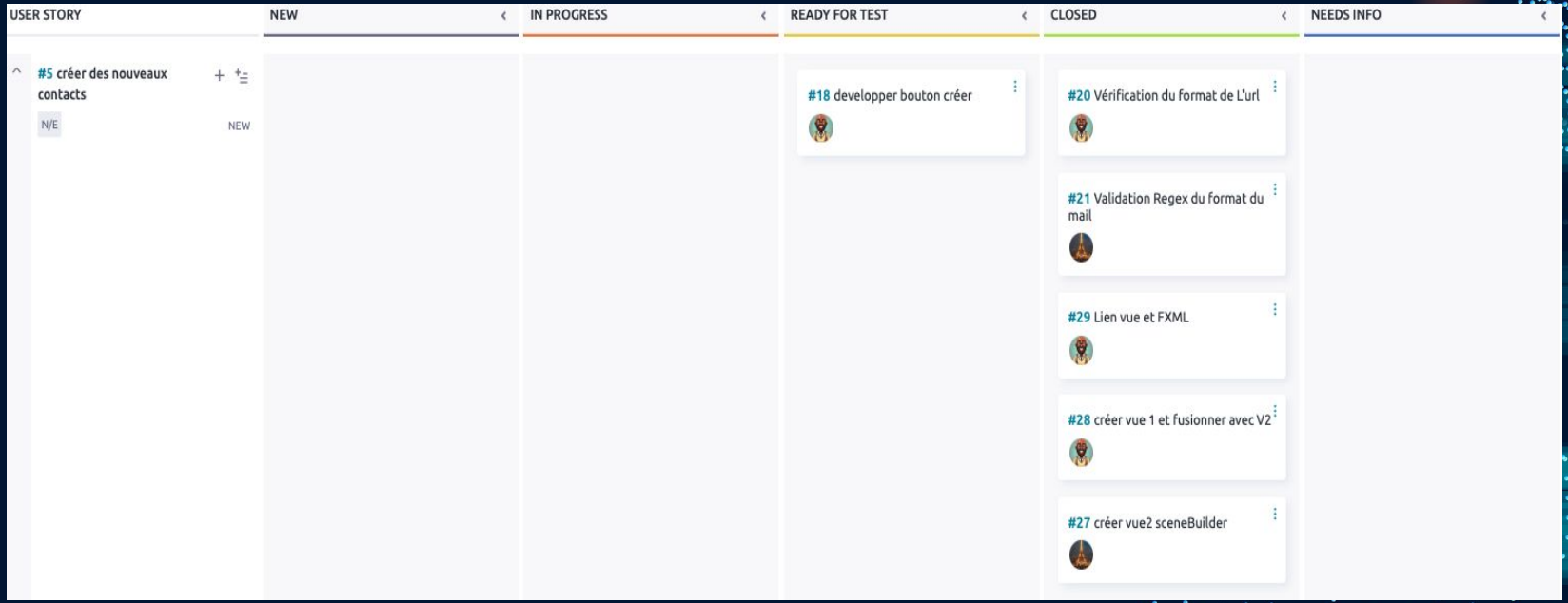
Example: Backlog avec création de User Stories qui sont rajoutés au Sprints



Construction d’un Sprint pour la création du Backlog et de User Stories

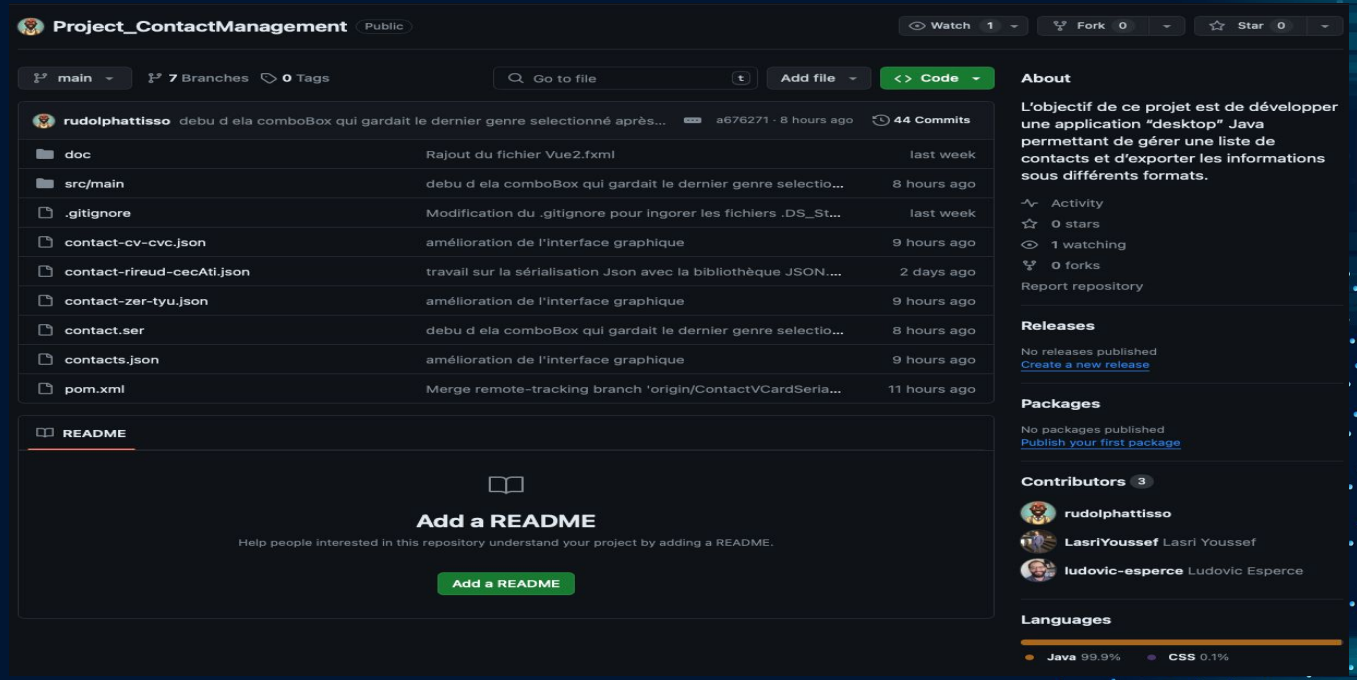
https://tree.taiga.io/project/rudolphattisso-contact_management/taskboard/conception-appliarchitecturediaguml

Example: Kanban



Mise en place du Dépôt Git partagé

Lien Git est privé mais vous avez l'honneur d'avoir un aperçu de la page principale.



3 Maquettage de l’application

- Création du Zoning et du Wireframe

Creation de Zoning et Wireframe

II.1. Gestion de contacts

L'application devra proposer une **interface graphique ergonomique** permettant de gérer une liste de contacts.

Les opérations attendues sont les opérations **CRUD** :

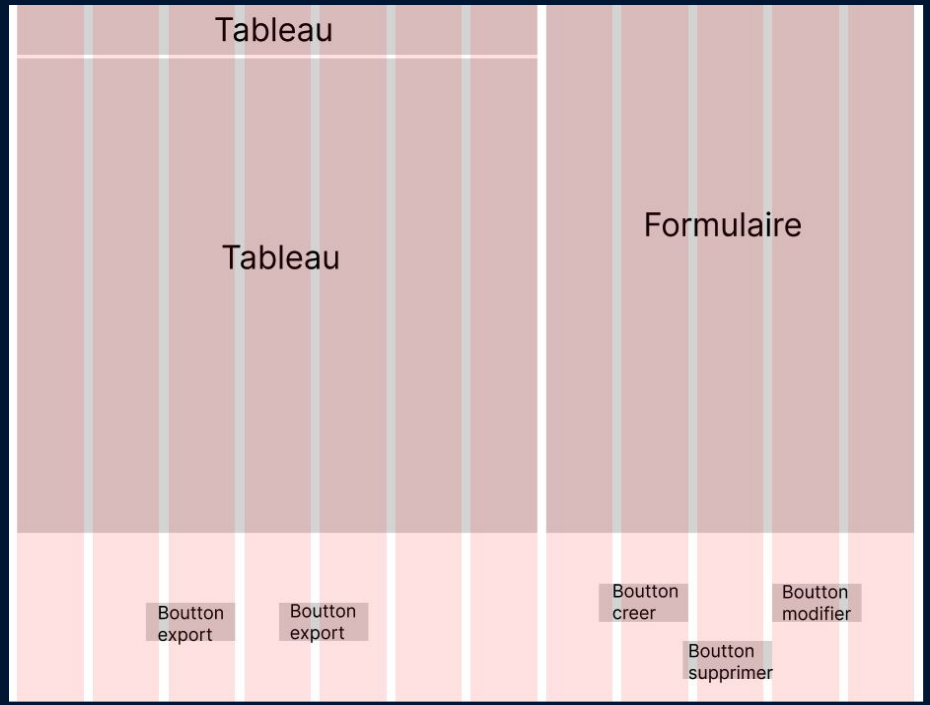
- consultations d'une liste de contact(s)
- création de contact(s)
- modification de contact(s)
- suppression de contact(s)

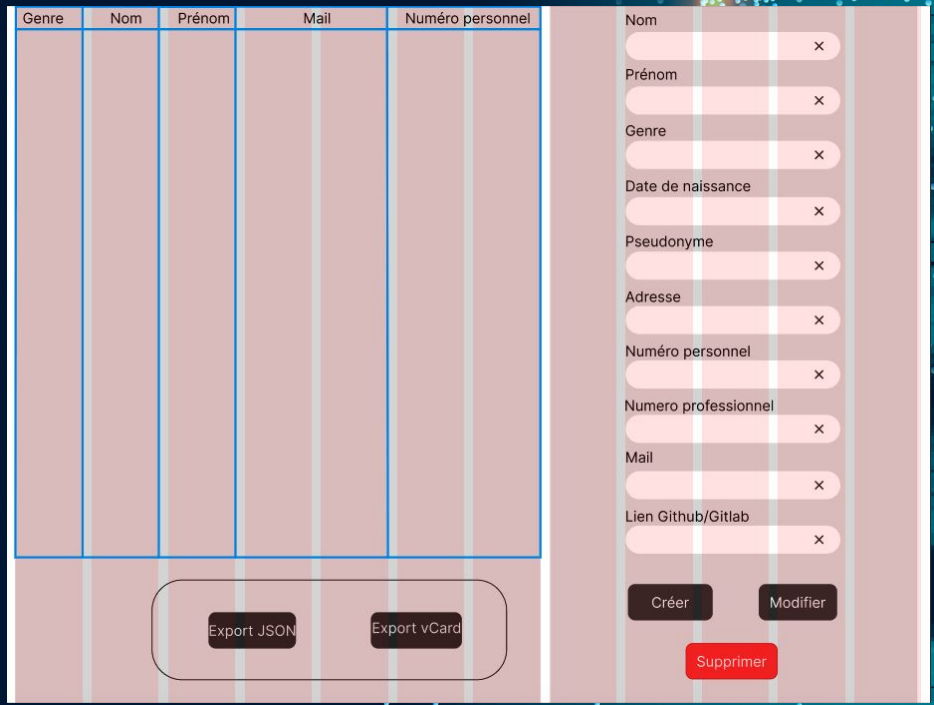
Une contact est défini par les caractéristiques suivantes :

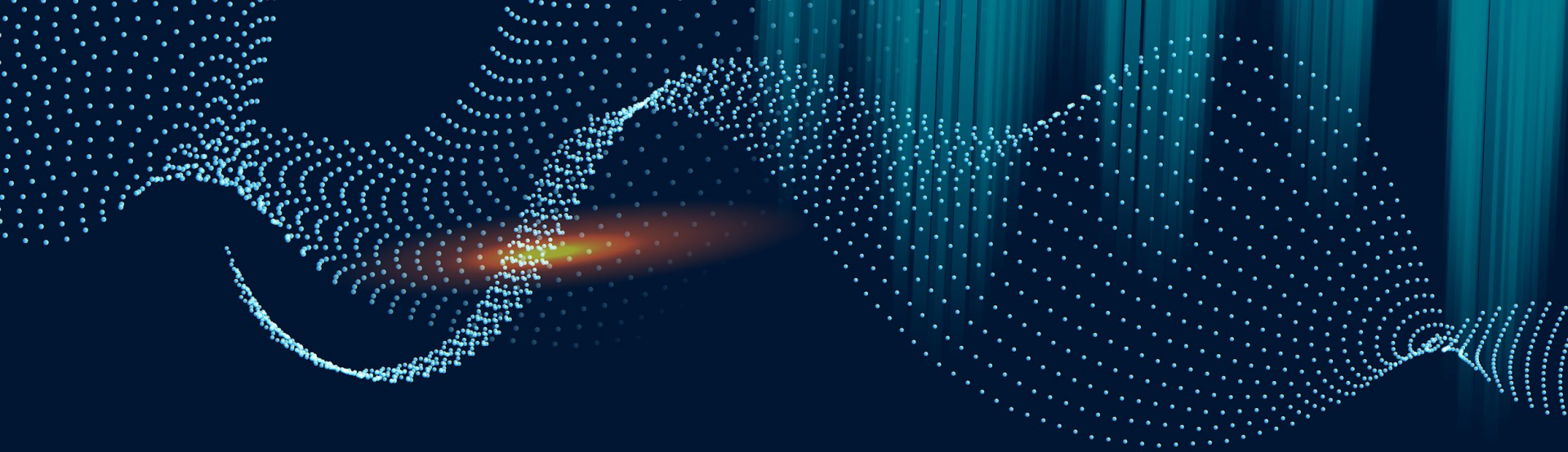
- **nom** (obligatoire)
- **prénom** (obligatoire)
- **genre** : homme/femme/non-binaire (obligatoire)
- **date de naissance** (optionnel)
- **un pseudonyme** (optionnel)
- **adresse** (obligatoire)
- **numéro de téléphone personnel** (obligatoire)
- **numéro de téléphone professionnel** (optionnel)
- **une adresse email** (obligatoire)
- **une adresse postale** (obligatoire)
- **un lien vers la page Github ou Gitlab du contact** (optionnel)

Creation de Zoning et Wireframe

Zoning et Wireframe adapaté sur deux formats: 1024px x 768px & 1920px x 1080px.





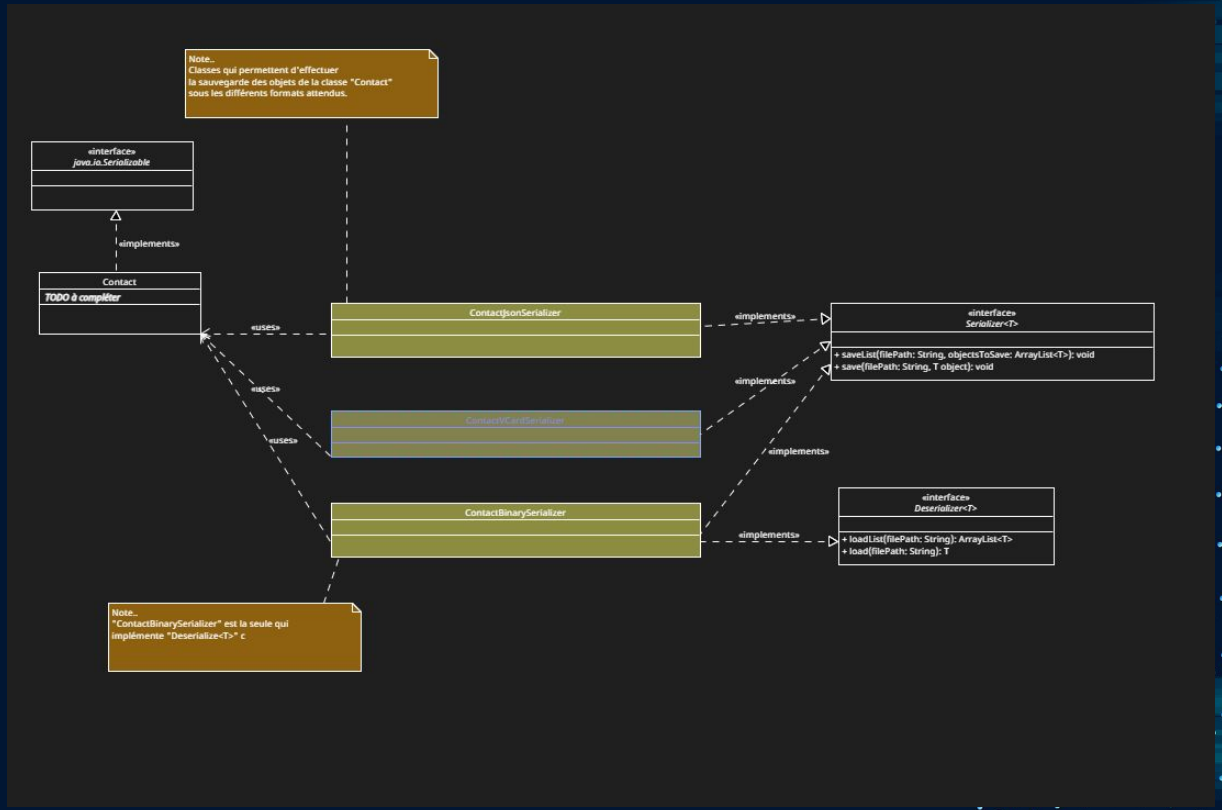


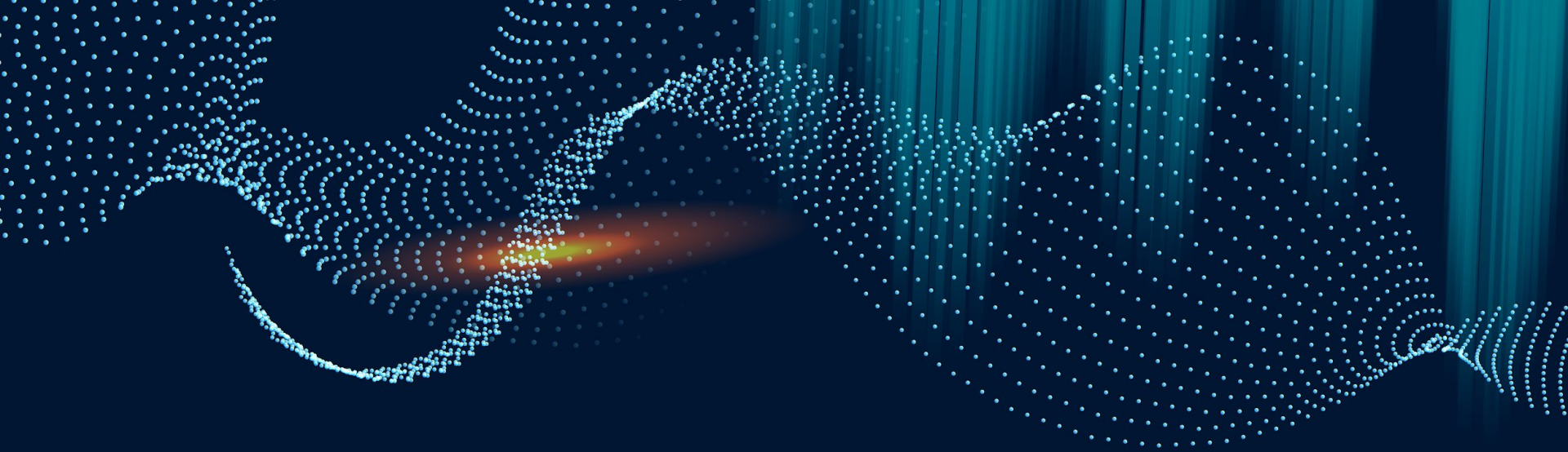
4 Conception de l’application

- Réflexion architecturale
- Conception de diagramme

Conception du diagramme

Diagramme uml initial

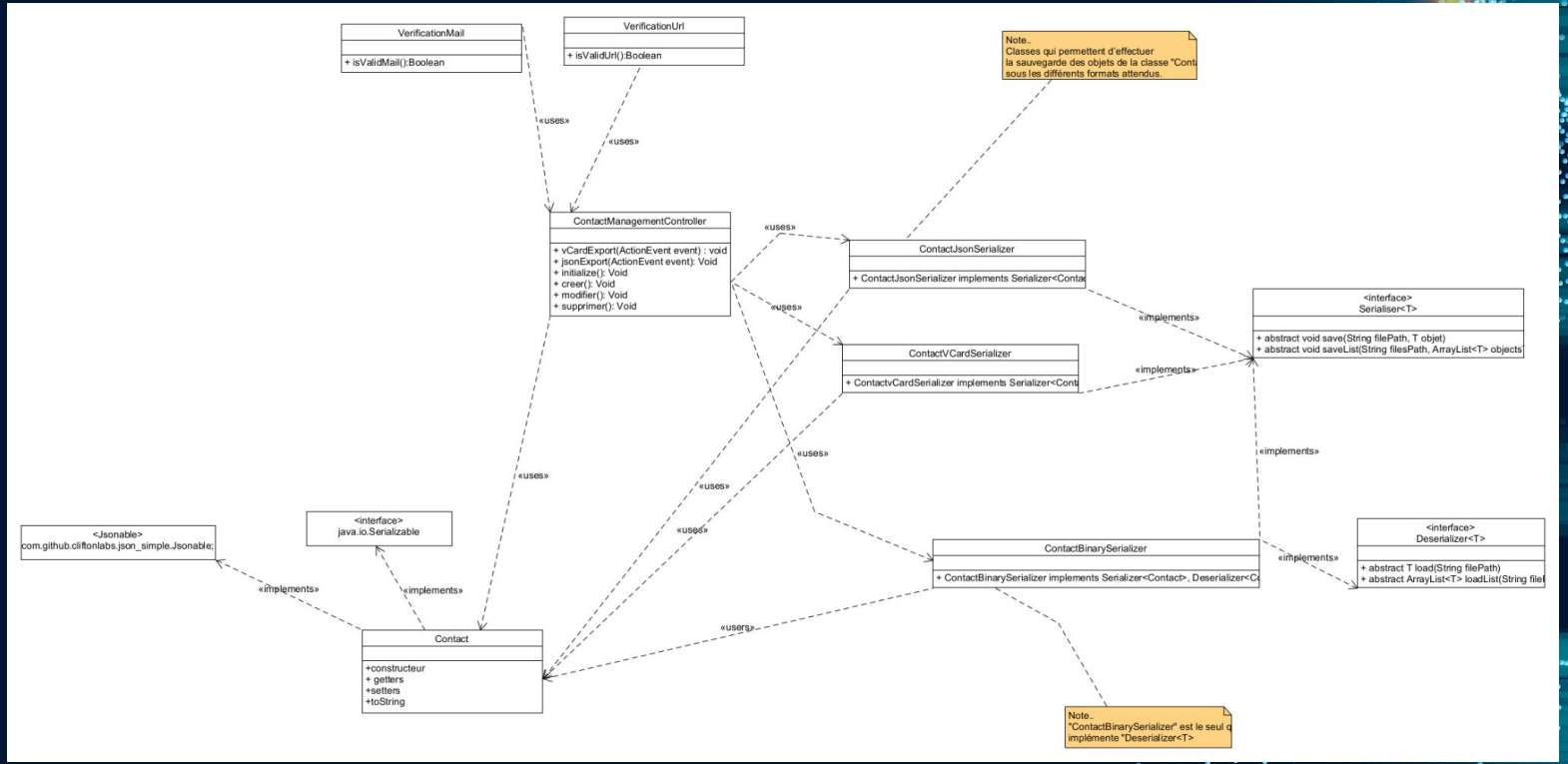




5 Début des développements

Conception du diagramme

Diagramme UML final



Conception du diagramme

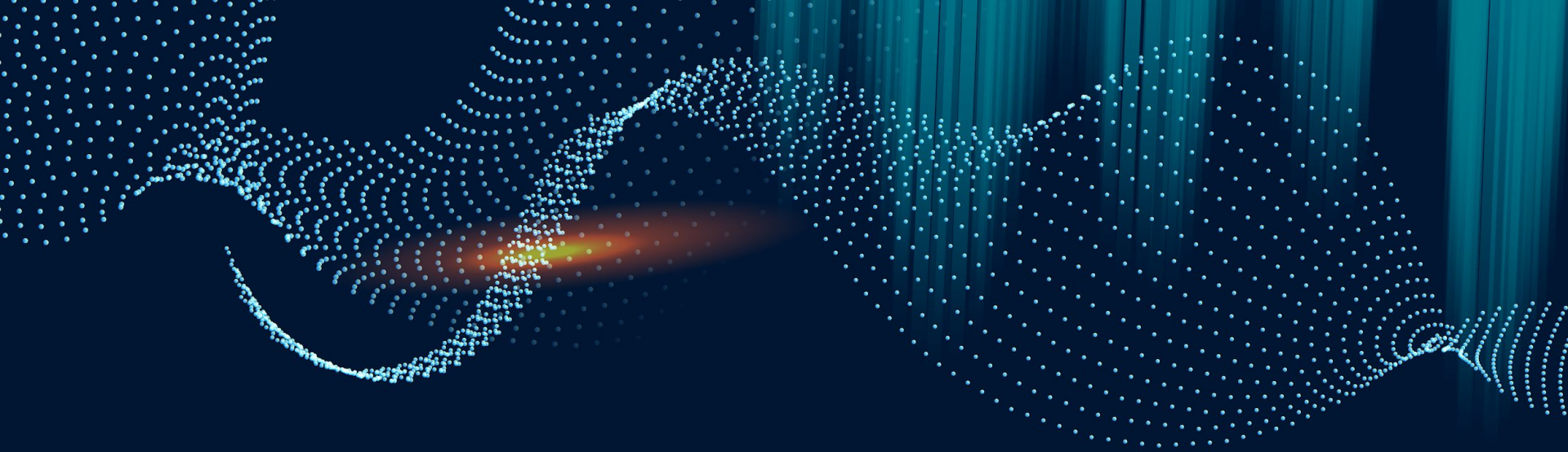
Veille OWASP: https://owasp.org/www-community/OWASP_Validation_Regex_Repository

```
<?xml version="1.0"?>

  <regex>
    <name>url</name>
    <pattern><![CDATA[^((((https?|ftps?|gopher|telnet|nntp)://)|(mailto:|news:))(%[0-9A-Fa-f]
{2}|[-()_.!~*';/?:@&=+$,A-Za-z0-9])+)([).!';/?:,][[:blank:|:blank:]])?$]]></pattern>
    <description>A valid URL per the URL spec.</description>
  </regex>
```

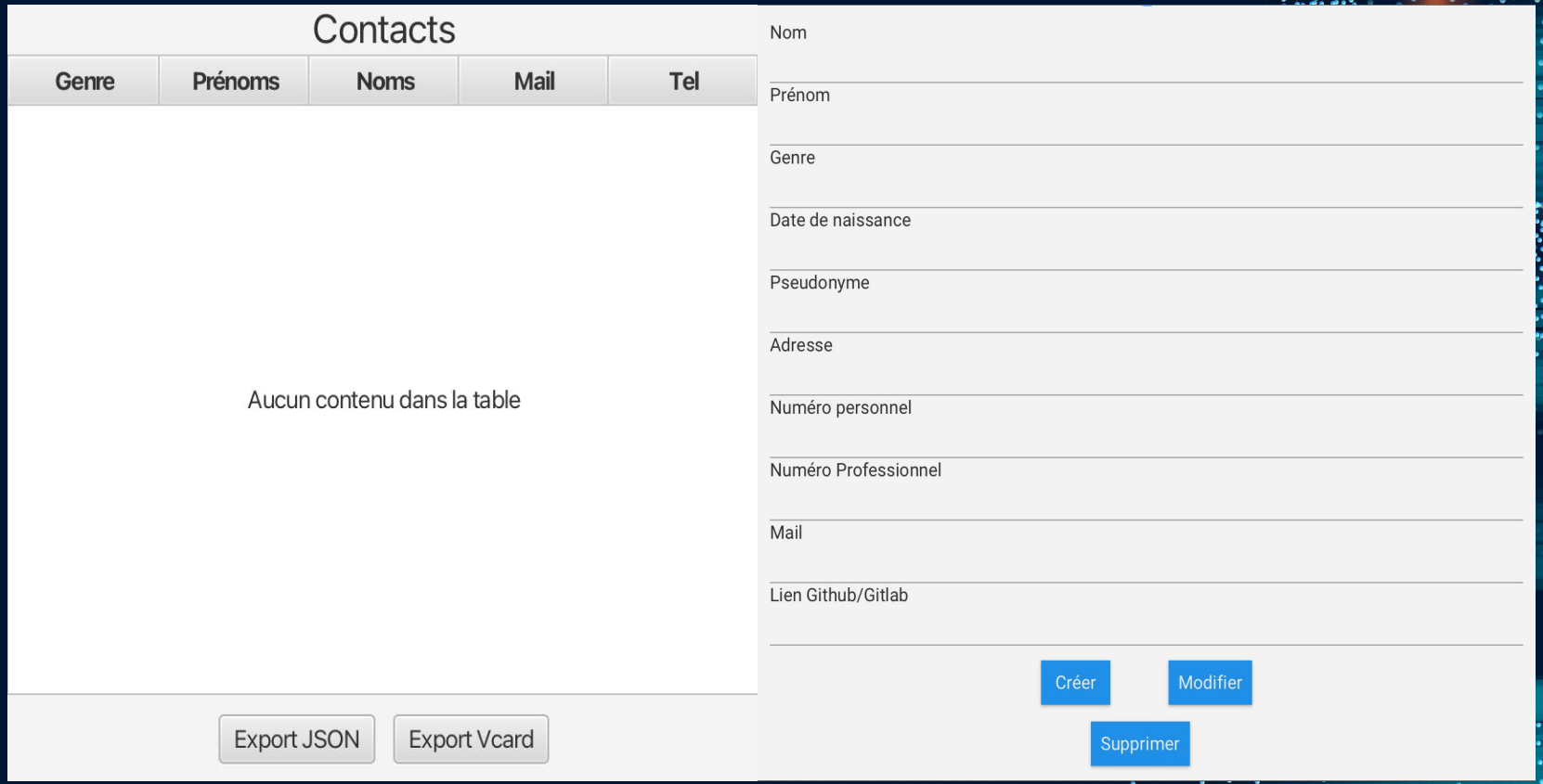
```
  <regex>
    <name>e-mail</name>
    <pattern><![CDATA[^[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]
{2,}$]]></pattern>
    <description>A valid e-mail address</description>
  </regex>

  <regex>
```

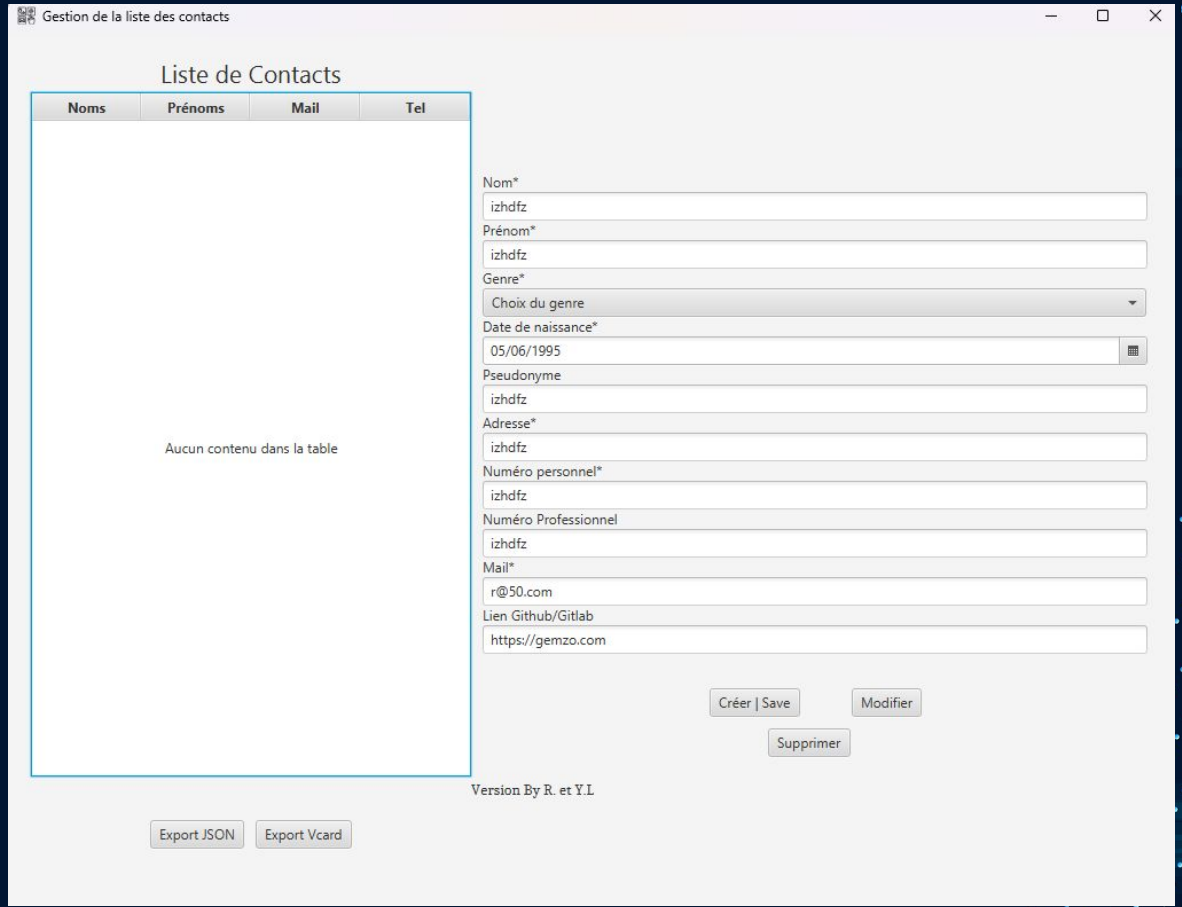


6 Tests fonctionnels du développement

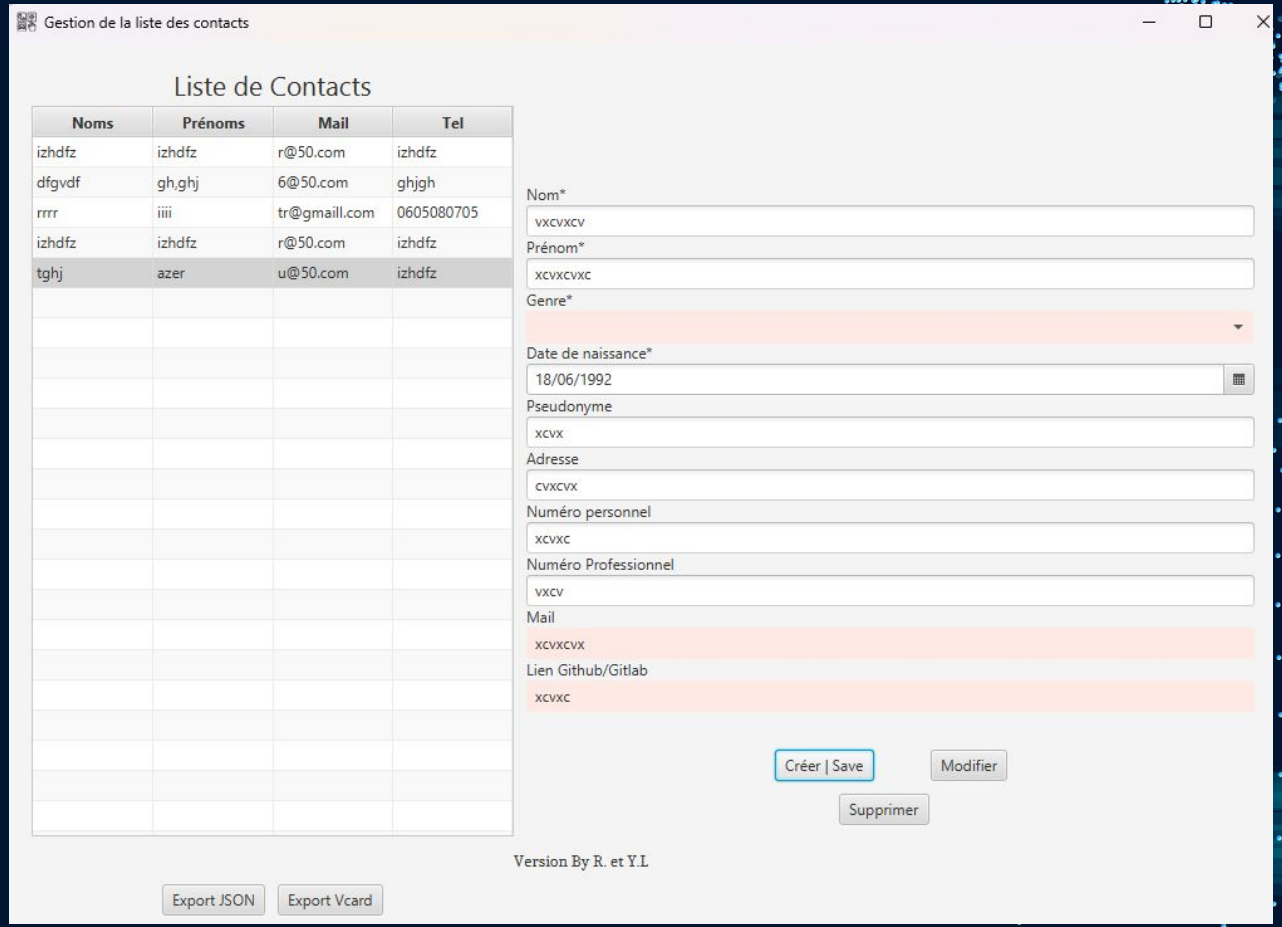
Vue initiale



Vue améliorée



Test fonctionnel du développement



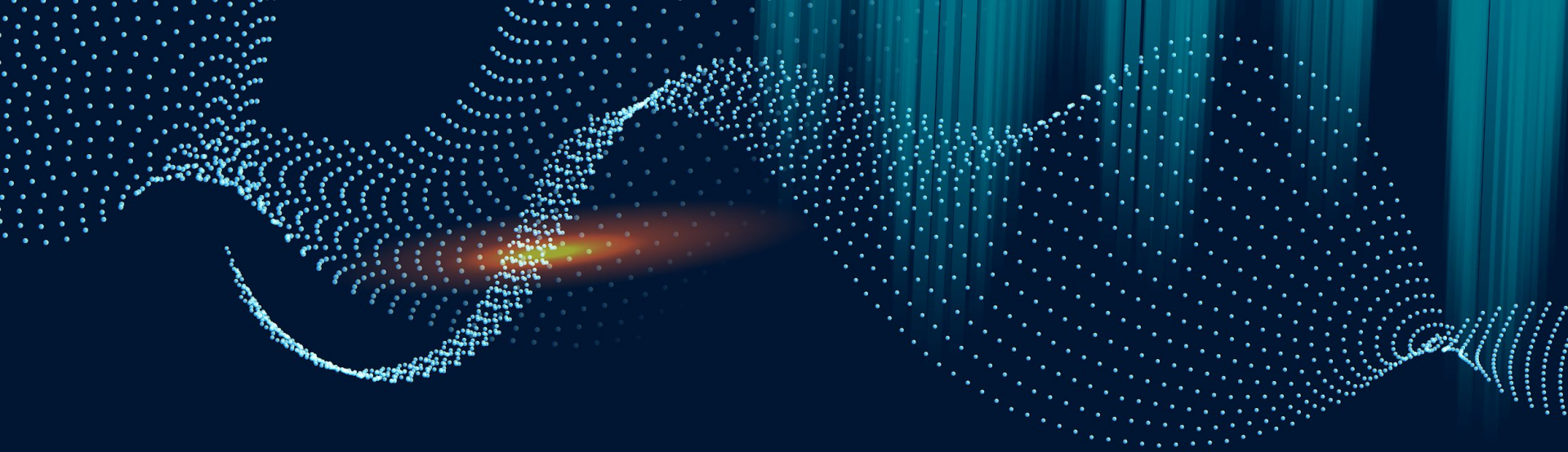
Test fonctionnel du développement

Test: Export JSON
Test: Export vCard

```json
{
    "mail": "o@gmail.com",
    "prénom": "tyu",
    "genre": "Homme",
    "adresse": "32 pal 41000",
    "date de naissance": "14/09/1994",
    "lienGit": "https://www.google.com/search?q=jfx+:how+to+get+the+value+of+combobox+display&sca_esv=b5919fffce7396f9&sxsrf=ADLYWILHsYyXNIDBPVWiJHaU8a02P98klw:1723019948651&ei=rDKzZs23J4yJkdUPyI-SkA0&ved=0
    "nom": "zer",
    "pseudo": "gu",
    "telPerso": "0608090804",
    "code Postale": "32 pal 41000"
}
```

Femmehjkhjkhj.vcf

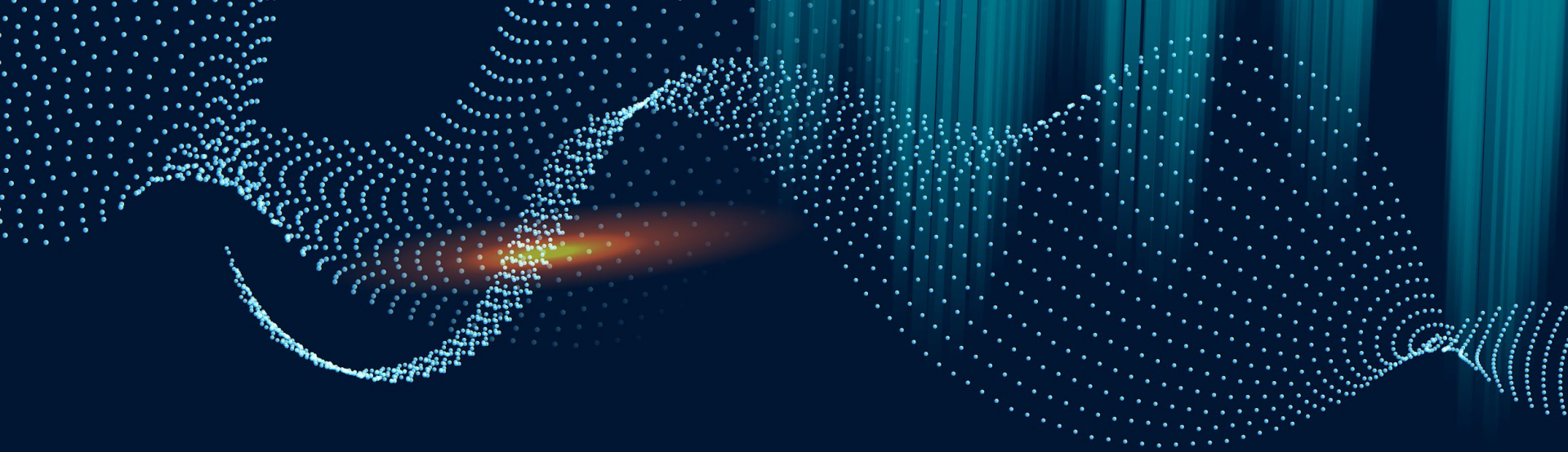
```
BEGIN:VCARD
VERSION:4.0
N:hjk;hjkhj;;Femme;
FN:hjkhj hjk
BIRTHDAY:sdd
TEL;TYPE=persnum, voice;VALUE=uridsd fsdfs
TEL;TYPE=worknum, voice; VALUE=urifsd
ADR;TYPE=HOME; PREF=1; LABELfsdfs
EMAIL:e@gmail.com
LINKGIT:https://www.radix-ui.com/colors
END:VCARD
```



7 Points d’améliorations

Quelques pistes d’amélioration

- Contraindre à l’aide des REGEX les champs nom prénom et num pro et perso afin d’avoir qu’ils ne contienne que les formats attendus
- Le datepicker devra se remettre à la date du jour après ajout d’une date de naissance.
- Ajouter un petit message quand le message a été créé et supprimé
- Logique de l’export Json : bug sur la selection(à moitier resolu)
- Ajout des Loggers aux points critiques susceptibles bloquer l’application
- Ajout du QRCode et export CSV;
- Inclure un barre de recherche.



8 Conclusion

- Livraison de l’application

Merci pour votre écoute