



DÉVELOPPEMENT INFORMATIQUE

RAPPORT DE STAGE

Application Web de gestion d'archives.

Réalisé par :

El cass Imad

Encadrée par :

Lamrani Kaoutar



Sommaire

- Remerciement
- Introduction
- Présentation de l'entreprise
- Présentation du project
- Les outils utilisés
- Le projet en détail
 - Étape 1 (La mise en page)
 - Étape 2 (Les routes)
 - Étape 3 (les pages)
 - Locaux et services
 - Beneficiaries
 - Rangers
 - Dossiers
 - Pieces
 - Étape 4 (Authentication)
 - Étape 5 (La base des données)



Remerciement

Je tiens à exprimer mes remerciements à toutes les personnes qui ont contribué d'une manière ou d'une autre à la réalisation de ce travail et à l'élaboration de ce document. A toutes ces personnes qui de près ou de loin me soutiennent et m'encouragent à aller de l'avant, je suis très reconnaissant à eux.



Introduction

Un stage est une expérience d'apprentissage professionnel qui offre un travail significatif et pratique lié au domaine d'études ou à l'intérêt professionnel d'un étudiant. Un stage donne à un étudiant la possibilité d'explorer et de développer sa carrière et d'acquérir de nouvelles compétences. Il offre à l'employeur la possibilité d'apporter de nouvelles idées et de l'énergie sur le lieu de travail, de développer les talents et potentiellement de créer un pipeline pour les futurs employés à temps plein.



Présentation de l'entreprise

GISMA IT SOLUTIONS est une agence de vente des logiciels, des solutions informatiques, des formations et conseils. Implantée à Agadir, Maroc.

Aussi il est spécialisée dans l'installation des réseaux informatiques et caméras de surveillance.



Présentation du projet

À cause d'une demande d'un client qui voulait gérer son archivage et le rendre informatisé et plus sécurisé.

On est commencé la création d'une application web de gestion d'archives pour faciliter la recherche, le stockage et la maintenance des archives.



Les outils utilisés

- React.js

React est une bibliothèque JavaScript pour créer des interfaces utilisateur.

- Laravel

Laravel est un framework php d'application Web avec une syntaxe expressive et élégante.

- MySQL

MySQL est un système de gestion de base de données relationnelle open-source.

- XAMP

XAMPP est un ensemble de logiciels permettant de mettre en place un serveur Web local.



Le projet en détail

pour faciliter la compréhension du project on va ajouter des images a chaque étape.

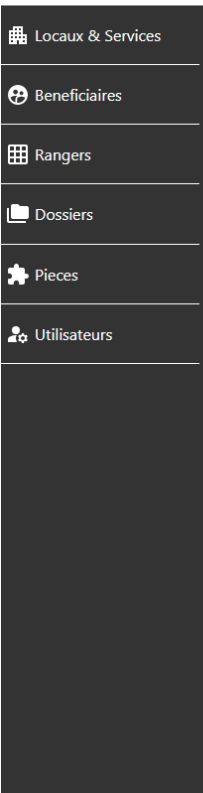
Étape 1 (La mise en page)

La premire etape est la mise en page de l'application.

<Layout></Layout>

Logo

Connexion



Étape 2 (Les routes)

Dans cette étape on va créer toutes les routes de l'application.

Le routage de l'application est créé avec react-router v6 (bibliothèque de routage de react js).

Ces routes doivent être définies dans le fichier *App.js*

```
App.js
18 const App = () => {
19   return (
20     <div className='App'>
21       <GlobalProvider>
22         <AlertProvider>
23           <BrowserRouter>
24             <Layout>
25               <AlertMsg>
26                 <Routes>
27                   <Route path='/' element={<Archive />} />
28                   <Route path='/archive' element={<Archive />} />
29                   <Route path='/beneficiere' element={<Beneficiere />} />
30                   <Route path='/dossiers' element={<Dossiers />} />
31                   <Route path='/Rangers' element={<Rangers />} />
32                   <Route path='/pieces' element={<Pieces />} />
33                   <Route path='/utilisateurs' element={<Users />} />
34                   <Route
35                     path='/dossiers/pieces-fornis'
36                     element={<PiecesFornis />}
37                   />
38                   <Route
39                     path='/dossiers/pieces-manquants'
40                     element={<PiecesManquants />}
41                   />
42                   <Route path='/login' element={<Login />} />
43                 </Routes>
44               </AlertMsg>
45             </Layout>
46           </BrowserRouter>
47         </AlertProvider>
48       </GlobalProvider>
49     </div>
50   );
51 };
52
53 export default App;
54
```

Ces routes permettent de naviguer vers différentes parties de l'application côté client. Il existe également d'autres types de routes côté serveur, c'est pour accéder aux données.

Ces routes doivent être définies dans le fichier *api.php*

```
api.php x
15
16 Route::post('/login', [UserController::class, 'login']);
17
18 Route::middleware('auth:sanctum')->group(function () {
19     Route::get('/beneficiaires', [BeneficiaireController::class, 'index']);
20     Route::get('/rangers', [RangerController::class, 'index']);
21     Route::get('/archive', [ArchiveController::class, 'index']);
22     Route::get('/dossiers', [DossierController::class, 'index']);
23     Route::get('/cellules', [CelluleController::class, 'index']);
24     Route::get('/typedossiers', [TypeDossierController::class, 'index']);
25     Route::get('/service', [ServiceController::class, 'index']);
26     Route::get('/pieces', [PieceController::class, 'index']);
27     Route::get('/typepieces', [TypePieceController::class, 'index']);
28 });
29
30 Route::middleware(['auth:sanctum', 'admin'])->group(function () {
31     Route::get('/users', [UserController::class, 'index']);
32     Route::post('/user/add', [UserController::class, 'create']);
33     Route::post('/archive/add', [ArchiveController::class, 'create']);
34     Route::post('/service/add', [ServiceController::class, 'create']);
35     Route::post('/beneficiaire/add', [BeneficiaireController::class, 'create']);
36     Route::put('/beneficiaire/update', [BeneficiaireController::class, 'update']);
37     Route::post('/ranger/add', [RangerController::class, 'create']);
38     Route::put('/ranger/update', [RangerController::class, 'update']);
39     Route::post('/typedossier/add', [TypeDossierController::class, 'create']);
40     Route::post('/dossier/add', [DossierController::class, 'create']);
41     Route::put('/dossier/update', [DossierController::class, 'update']);
42     Route::post('/piece/add', [PieceController::class, 'create']);
43     Route::put('/piece/update', [PieceController::class, 'update']);
44     Route::post('/page_piece/add', [PagePieceController::class, 'create']);
45     Route::post('/typepiece/add', [TypePieceController::class, 'create']);
46 });
47
48 Route::middleware(['auth:sanctum', 'admin', 'master'])->group(function () {
49     Route::delete('/dossier/destroy/{id}', [DossierController::class, 'destroy']);
50     Route::delete('/piece/destroy/{id}', [PieceController::class, 'destroy']);
51     Route::delete('/user/destroy/{id}', [UserController::class, 'destroy']);
52     Route::delete('/archive/destroy/{id}', [ArchiveController::class, 'destroy']);
53     Route::delete('/service/destroy/{id}', [ServiceController::class, 'destroy']);
54     Route::delete('/beneficiaire/destroy/{id}', [BeneficiaireController::class, 'destroy']);
55     Route::delete('/ranger/destroy/{id}', [RangerController::class, 'destroy']);
56 });
57
```

Étape 3 (les pages)

Les Pages de l'application :

- Locaux et services
- beneficiaries
- Rangers
- Dossiers
- Pieces
- Utilisateurs

Dans toutes les pages, l'utilisateur peut voir, mettre à jour, supprimer ou créer.











Pour rendre tout cela facile, nous utiliserons une bibliothèque de React pour l'affichage en grille appelée ag-grid.


```
<AgGridReact
  ref={gridRef}
  columnDefs={columns}
  rowData={users}
  defaultColDef={defaultColDef}
  rowSelection='single'
  editType='fullRow'
  suppressClickEdit
  stopEditingWhenCellsLoseFocus
/>
```

Locaux et services


Un Local est un conteneur global du projet.

Les services seront utilisés dans la page Utilisateurs, pour ajouter chaque utilisateur à son service.

Locaux			Services		
Code	Intitulé	Action	Code	Intitulé	Action
l1	local 1	 	s1	service 1	 
l2	local 2	 	s2	service 2	 
l3	local 3	 			

Le colonne Action est ajouter dans to les grilles.
au click sur le button  une fonction d'ajoute ou de modification va se déclencher.



```
const setArchive = async () => {
  props.api.stopEditing(false);
  try {
    const req = await axios.post(`/archive/add`, props.data);
    const data = await req.data;
    // execute alert
    showAlert(() => {
      return {
        state: true,
        text: data.msg,
        severity: data.severity,
      };
    });
  } catch (error) {
    console.log('error', error);
  }
};
```

Au click sur le button  une fonction de suppression va se déclencher.

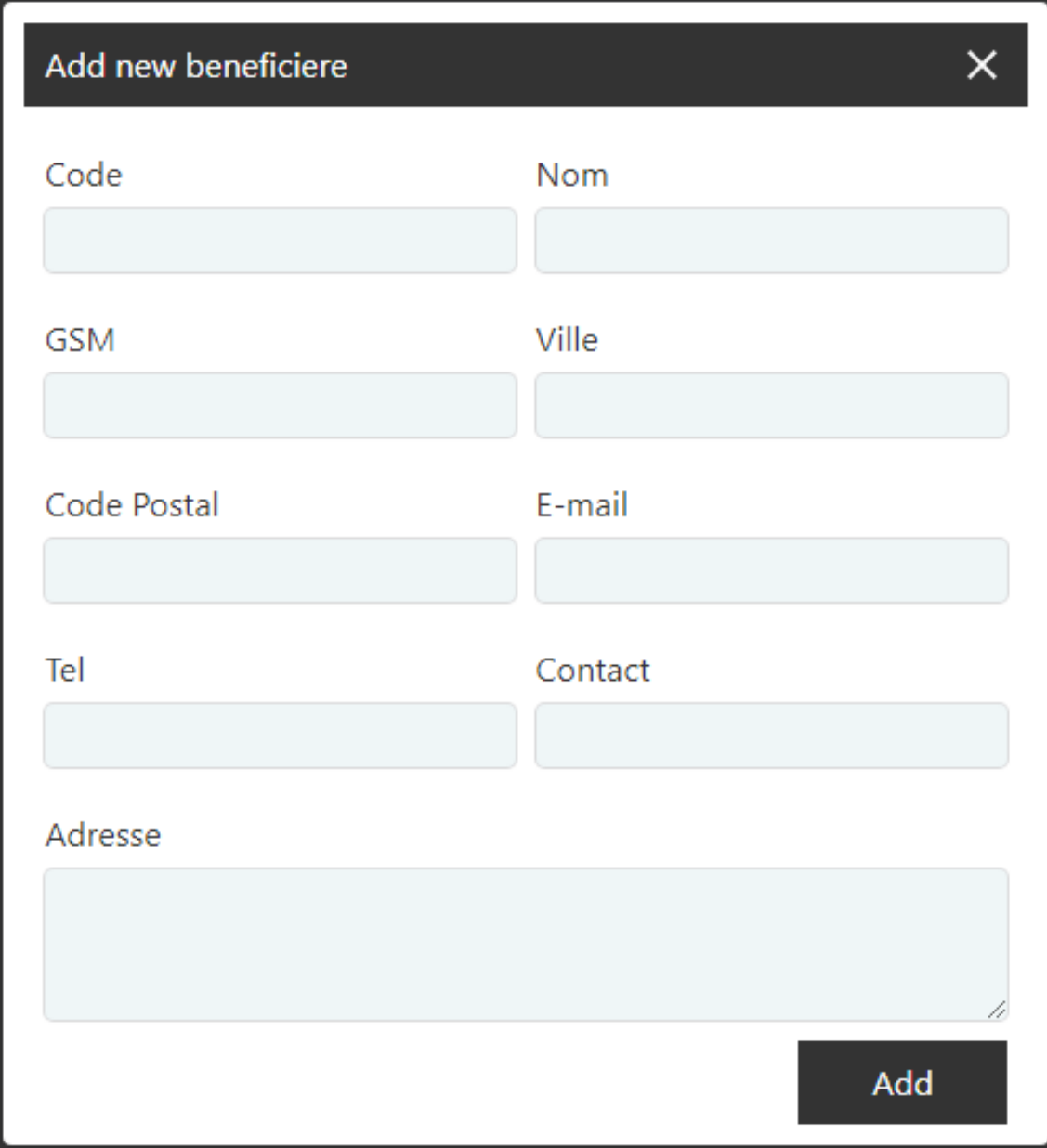
```
const deleteArchive = async () => {
  try {
    const req = await axios.delete(`/archive/destroy/${props.data.id}`);
    const data = await req.data;
    // execute alert
    showAlert(() => {
      return {
        state: true,
        text: data.msg,
        severity: data.severity,
      };
    });
    if (data.success) {
      // if delete succeeded in db => delete the row in client side
      const selectedData = gridRef.current.api.getSelectedRows();
      gridRef.current.api.applyTransaction({ remove: selectedData });
      gridRef.current.api.refreshCells({ force: true });
    }
  } catch (error) {
    console.log(error);
  }
};
```

Beneficiaries

Beneficiaires

Code	Nom	Adre...	Ville	CP	Email	Tel	Cont...	GSM	Action
b1	benef 1	rue	agadir	86150	benef@...	058796...	058796...	058796...	 

Au click sur le button  une boîte s'affichera pour ajouter un beneficiarie.



A modal form titled "Add new beneficiere" with a close button (X) in the top right corner. The form contains several input fields arranged in two columns. The first column includes fields for "Code", "GSM", "Code Postal", and "Tel". The second column includes fields for "Nom", "Ville", "E-mail", and "Contact". At the bottom left, there is a large text area labeled "Adresse". At the bottom right, there is a dark button labeled "Add".

Code	Nom
<input type="text"/>	<input type="text"/>
GSM	Ville
<input type="text"/>	<input type="text"/>
Code Postal	E-mail
<input type="text"/>	<input type="text"/>
Tel	Contact
<input type="text"/>	<input type="text"/>
Adresse	
<input type="text"/>	
<input type="button" value="Add"/>	


Au click sur le button Ajouter la fonction `addNewBeneficiaire()` va se déclencher.

```
const addNewBeneficiere = async () => {
  try {
    const req = await axios.post(`${baseUrl}/beneficiaire/add`, beneficiaire);
    const data = await req.data;
    if (data.success) {
      setDisplay(false);
      gridRef.current.api.applyTransaction({ add: [data.beneficiaire] });
      gridRef.current.api.refreshCells({ force: true });
      //execute alert
      showAlert(() => {
        return {
          state: true,
          text: "Le beneficiaire est bien ajouter.",
          severity: "success",
        };
      });
    }
  } catch (error) {
    setDisplay(false);
    showAlert(() => {
      return {
        state: true,
        text: "Le beneficiaire est pas ajouter.",
        severity: "error",
      };
    });
    console.log(error);
  }
};
```

Rangeres


Rangeres


+ 

code	intitul	Archive	nbrLignes	nbrColonnes	Action
r1	ranger 1	local 1	1	1	 

Dossiers

Dossiers											
Add new type											+
Num	Type	Service	Beneficiaire	Date	Cellule	Annee	Objet	Disponible	Valider	Valider Par	Action
doss1	type doss 1	service 1	benef 1	2022-02-15	r1-1-1	2020	object dossier	none	none		 

Au click sur le button  une boîte s'affichera pour ajouter un dossier.

Add new Dossier 

Numero

Type

type doss 1


Service

s1

Beneficiaire

benef 1

Date


mm/dd/yyyy 

Cellule


Annee

Objet

Disponible



Valider



Add

```
const addNewDossier = async () => {
  try {
    const req = await axios.post(`/dossier/add`, dossier);
    const data = await req.data;
    setDisplay(false);
    setAlert(() => {
      return {
        state: true,
        text: data.msg,
        severity: data.severity,
      };
    });
  } catch (error) {
    console.log(error);
  }
};
```


Pieces

Les piece se sont le contenu de l'archivage.

Pieces

Num	Type	Intitule	Dossier	Action
p1	tp1	piece 1	doss1	<div><div></div><div></div></div>
p2	tp2	piece 2	doss1	<div><div></div><div></div></div>

Le fichier de piece

Table of Contents

Introduction

4

Study Guide Purpose and Format

5

Reading Schedule

6

Session One

SOMETHING IS MISSING

8

Session Two

LIVING SOULFULLY

12

Session Three

FALLING IN LOVE

16

Session Four

HUNGRY

20

Session Five

THE SECRET TO EXCELLENCE

24

Session Six

LET YOUR LIGHT SHINE

28

<

>

<<

<

>

>>

Page 3 of 32

Pour ajouter une piece, un fichier pdf doit être choisi

Ajouter une piece

Numéro

Intitule

Type

Dossier

Fichier

Add

Utilisateurs

Utilisateurs					
					+
Nom	Email	Mot de pass	Type	Service	Action
user	user@mail.com	*****	user	service 1	 
admin	admin@mail.c...	*****	admin	service 1	 
master	master@mail.c...	*****	master	service 1	 

Pour des raisons de sécurité, l'application est deviser a trois type d'utilisateurs.

- **user**

Peut seulement voir les données.

- **admin**

Peut voir, mettre à jour, créer, mais il ne peut pas supprimer.

- **master**

Peut tout faire.

NB: Le master est le seul qui peut voir les utilisateurs

Cette partie est discuter dans l'étape d'authentification.

Étape 4 (Authentication)

Cette partie est très cruciale pour que l'application fonctionne parfaitement.

Une fois l'utilisateur connecté, un **token** sera généré pour détecter le type d'utilisateur et ce qu'il peut faire avec les données.

token: "76|jeMMYUjiby1dQFAQUQOMr5YRE3VXgp7YVZGOFmnN"

En suite le token sera stocké pour faire des requêtes au serveur.

Les routes de serveurs sont protégé par des middleware.

Les middleware permettent de filtrer les requêtes HTTP entrant dans votre application.

Ces middleware sont défini dans le fichier **Karnel.php**.

```
protected $routeMiddleware = [  
    'auth' => \App\Http\Middleware\Authenticate::class,  
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  
    'can' => \Illuminate\Auth\Middleware\Authorize::class,  
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  
    'admin' => \App\Http\Middleware\AdminMiddleware::class,  
    'master' => \App\Http\Middleware\MasterMiddleware::class,  
];
```

Étape 5 (La base des données)

Dans cette étape on va voir le shema de la base de données.

