Projet NonoLeRobot

07/03/2023

Collaborateurs

- **Chef de projet :**
  Nicolas Marchand
- **Apprenti-Dev :**
  Alexia LABARTHE
  Mathys PAQUEREAU

Cahier des charges

Création d'une API type « ChatBot »

Objectif :

- L'intégrer à LoGeAs-FullWeb
- 2 interfaces
- Base de connaissance

Utilisateur

Pose sa question et obtient une réponse sans passer par l'assistance

Assistance

Ajoute ou modifie des réponses dans la base de connaissance

Disponible à tous les utilisateurs LoGeAs-FullWeb

Cahier des charges

Fonctions accessibles sans identifications

- Suite à la saisie d'une Question :
  - Proposer une suite de Réponses
  - Classées par pertinence
  → Suite de titres cliquables
- Si pas de résultat ou pas de réponses insuffisantes → envoi mail à l'assistance
- Mot clés non-connus → ajout dans la BDD
- Mot clés incrémentés à chaque demande
- Réponses incrémentés à chaque ouverture
- Aucune gestion de droits demandé

- **Fonctions accessibles avec identifications**
- Saisie d'un mot de passe en dur (V1)
- Créer/rechercher un document
- Le modifier
- Le lier/délier avec des mots clés
- Le lien avec un lien

Cahier des charges

Aspects techniques

BACK	FRONT
« Free Pascal Lazarus » complété de l'ORM mORMot	FramWork Angular Bibliothèque DevExtreme

Les documents seront au format HTML stocké dans la base de donnée

Back-End

Exemple avec procédure Recherche

1) Déclaration de la procédure :

```
  .  //procedure de recherche  Question -> Reponses
40     procedure Recherche(Ctxt: TSQLRestServerURIContext); //http://localhost:8087/root/Recherche?IdTypeBase=13&Question=Comment imputer les salaires?
```

Back-End

Exemple avec procédure Recherche

2) Mise en place de la procédure :

```
procedure TNonoServer.Recherche(Ctxt: TSQLRestServerURIContext);
Var
   Question:String;
   IdTypeBase:Integer;
   ListeIDMot,ListeIDSoundex:String;
   Res:String;
   SQL:RawUTF8;
begin
  Question:=Ctxt.InputUTF8['Question'];
  IdTypeBase:=Ctxt.InputIntOrVoid['IdTypeBase'];
  writeln('Question :'+Question);
  writeln('IdTypeBase :'+intToStr(IdTypeBase));

  if length(question)<2 then
  begin
    Ctxt.Returns('{}');
   exit;
  end;

   if Question[1]='=' then
    begin
      Question := Trim(Copy(Question,2,MaxInt));
      if Question[1]='*' then
        begin
          Res:=GetJsonFromSQL('SELECT * FROM Reponse');
        end
      else Res:=GetJsonFromSQL(formatUTF8('SELECT * FROM Reponse Where ID=?',[],[Question]));
    end
  else
    begin
      ListeIDMot:=MotsToListeID(Question,False,False);
      ListeIDSoundex:=MotsToListeIDSoundex(Question);

      SQL:=FormatUTF8(
        'SELECT Max(Pertinence), ID, Titre, Text, IDTypeBase FROM ( '+
           '  SELECT Count(*) as Pertinence, L.IdReponse as ID, R.Titre, R.Text, R.IdTypeBase '+
           '  FROM Lien L, Reponse R '+
           '  WHERE (L.IDMotClef in %) and (R.ID=L.IdReponse) '+
           '  GROUP BY IdReponse '+
          'UNION '+
           '  SELECT Count(*)*.5 as Pertinence, L.IdReponse as ID, R.Titre, R.Text, R.IdTypeBase '+
           '  FROM Lien L, Reponse R '+
           '  WHERE (L.IDMotClef in %) and (R.ID=L.IdReponse) '+
           '  GROUP BY IdReponse '+
        ')',[ListeIDMot,ListeIDSoundex],[]);

      if IdTypeBase=0 then
        SQL:=SQL+' GROUP BY ID ORDER BY Pertinence DESC'
       else
        SQL:=SQL+FormatUTF8('WHERE ((IdTypeBase=?) or (IdTypeBase=?) or (IdTypeBase is null)) GROUP BY ID ORDER BY Pertinence DESC',[],[0,IdTypeBase]);

      Res:=GetJsonFromSQL(SQL);
    end;
  Ctxt.Returns(Res);
end;
```

Back-End

Exemple avec procédure Recherche

3) Utilise les fonctions suivantes :

- GetJsonFromSQL

```
{
 Rend le résultat de la requête SQL au format JSON
 @param (RawUTF8) Requête SQL
 @returns Rend le résultat de la requête SQL au format JSON
}
function TNonoServer.GetJsonFromSQL(SQL:RawUTF8): RawUTF8;
var
  Table:TSQLTableJSON;
  Doc:Variant;
begin
  Table:=Self.ExecuteList([],SQL);
  Try
    Result:='';
    If Table.RowCount=0 then
     Result:='{}'
     else begin
       Table.ToDocVariant(doc,True);
       result:=VariantToUTF8(Doc);
     end;
  finally
    { Quoi qu'il arrive entre le Try et le finally, je libère la mémoire et je passe entre le finally et le end }
    Table.free;
  end;
end;
```

Back-End

Exemple avec procédure Recherche

```
procedure TNonoServer.Recherche(Ctxt: TSQLRestServerURIContext);
Var
   Question:String;
   IdTypeBase:Integer;
   ListeIDMot,ListeIDSoundex:String;
   Res:String;
   SQL:RawUTF8;
begin
  Question:=Ctxt.InputUTF8['Question'];
  IdTypeBase:=Ctxt.InputIntOrVoid['IdTypeBase'];
  writeln('Question :'+Question);
  writeln('IdTypeBase :'+intToStr(IdTypeBase));

  if length(question)<2 then
  begin
    Ctxt.Returns('{}');
   exit;
  end;

   if Question[1]='=' then
    begin
      Question := Trim(Copy(Question,2,MaxInt));
      if Question[1]='*' then
        begin
          Res:=GetJsonFromSQL('SELECT * FROM Reponse');
        end
      else Res:=GetJsonFromSQL(formatUTF8('SELECT * FROM Reponse Where ID=?',[],[Question]));
    end
  else
    begin
      ListeIDMot:=MotsToListeID(Question,False,False);
      ListeIDSoundex:=MotsToListeIDSoundex(Question);

      SQL:=FormatUTF8(
        'SELECT Max(Pertinence), ID, Titre, Text, IDTypeBase FROM ( '+
           '  SELECT Count(*) as Pertinence, L.IdReponse as ID, R.Titre, R.Text, R.IdTypeBase '+
           '  FROM Lien L, Reponse R '+
           '  WHERE (L.IDMotClef in %) and (R.ID=L.IdReponse) '+
           '  GROUP BY IdReponse '+
           'UNION '+
           '  SELECT Count(*)*.5 as Pertinence, L.IdReponse as ID, R.Titre, R.Text, R.IdTypeBase '+
           '  FROM Lien L, Reponse R '+
           '  WHERE (L.IDMotClef in %) and (R.ID=L.IdReponse) '+
           '  GROUP BY IdReponse '+
        ')',[ListeIDMot,ListeIDSoundex],[]);

      if IdTypeBase=0 then
        SQL:=SQL+' GROUP BY ID ORDER BY Pertinence DESC'
       else
        SQL:=SQL+FormatUTF8('WHERE ((IdTypeBase=?) or (IdTypeBase=?) or (IdTypeBase is null)) GROUP BY ID ORDER BY Pertinence DESC',[],[0,IdTypeBase]);

      Res:=GetJsonFromSQL(SQL);
    end;
  Ctxt.Returns(Res);
end;
```

3) Utilise les fonctions suivantes :

- MotsToListeID

Front-End

Exemple avec procédure Recherche

1) Création du fichier nono.service.ts

→ On déclare la propriété urlBase qui est l'URL d'accès à notre serveur local

```
15    //URL locale
16    public urlBase:string ='http://localhost:8087/root/';
```

→ On déclare la méthode Recherche

```
28  public Recherche(IdTypeBase:number, Question:string): Observable<any> {      Mathys, il y a 17 heures
29  let URL = this.urlBase + 'Recherche?IdTypeBase='+IdTypeBase+'&Question='+encodeURIComponent(Question);
30  console.log('URL : ',URL);
31    return this.http.get(URL)
32      .pipe(
33        retry(2),
34        catchError(err => this.getError(err))
35      );
36  }
```

Front-End
Exemple avec procédure Recherche

→ On gère les erreurs potentielles avec la méthode getError

```
108   getError(error: any) {
109     let message = '';
110     if (error.error instanceof ErrorEvent) {
111       // gère les erreurs côté client
112       message = `Error: ${error.error.message}`;
113     } else {
114       // gère les erreurs côté serveur
115       message = `Error Code: ${error.status}\nMessage: ${error.message}`;
116     }
117     console.log('getError ', message);
118     return throwError(error);
119   }
```

Front-End

Exemple avec procédure Recherche

→ Implémentation et utilisation de la méthode recherche du back

Dans le fichier assistance-pop-up.component.ts

→ **Import des modules nécessaires**

```
1   import { Component } from '@angular/core';
2   import { NonoService, TSQLReponse, TSQLReponses } from '../nono.service';
3   import notify from 'devextreme/ui/notify';
```

→ Composant text-box de la librairie DevExtreme du côté HTML

```
16    <dx-text-box
17      [(value)]="Question"
18      placeholder="Saisissez votre demande"
19      valueChangeEvent="keyup"
20      (onValueChanged)="onCheckboxValueChanged($event)"
21      [maxLength]="40">
22    </dx-text-box>
```

Front-End

Exemple avec procédure Recherche

→ Côté TS du composant text-box et de son évènement onValueChanged

```
36    onCheckboxValueChanged(e:any) {
37      this.nono.Recherche(this.IdTypeBase, this.Question).subscribe({
38        next: (res: any) => {console.log('ICI ',res); this.Reponses=res},
39        error: (error) => {console.log('ERR ICI ',error);},});
40    }       Mathys, il y a 5 jours • commit2 …
```

→ Affichage des réponses récoltés

Composant grille de DevExtreme pour les réponses

```
70
71  <dx-data-grid id="gridContainer" [dataSource]="Reponses" keyExpr="ID" [columnAutoWidth]="true" [showBorders]="
72    [focusedRowEnabled]="true" [(autoNavigateToFocusedRow)]="autoNavigateToFocusedRow"
73    (onFocusedRowChanged)="VoirReponse($event)" *ngIf="isConnected">
74
75    <dxi-column dataField="ID" [visible]="isConnected"></dxi-column>
76    <dxi-column dataField="Titre"></dxi-column>
77  </dx-data-grid>
```

Front-End

Exemple avec procédure Recherche

→ Côté TS du composant data-grid et de son évènement onFocusedRowChanged

```
42  onFocusedRowChanged(e:any) {
43    this.ligneCourante = e.rowIndex;
44    this.ReponseCourante = this.Reponses[this.ligneCourante];
45    console.log("Avant changement",e.rowIndex)
46    this.nono.GetMotClef(this.ReponseCourante.ID).subscribe({
47      next: (res: any) => {console.log('ICI ',res); this.motsclefs = res.Liste},
48      error: (error) => {console.log('ERR ICI ',error);},
49    })
50    console.log("Mots Clefs :", this.motsclefs)
51  }
```

Front-End

Exemple avec procédure Recherche

```
16    title = 'nono-angular';
17    Question = '';
18    Reponses! : TSQLReponses;
19    ReponseCourante!:TSQLReponse;
20    motsclefs = '';
21    autoNavigateToFocusedRow = true;
22    ligneCourante: any;
23    isAssistance : boolean = false
24    IdTypeBase : number = 0;
```

→ Variable et classe

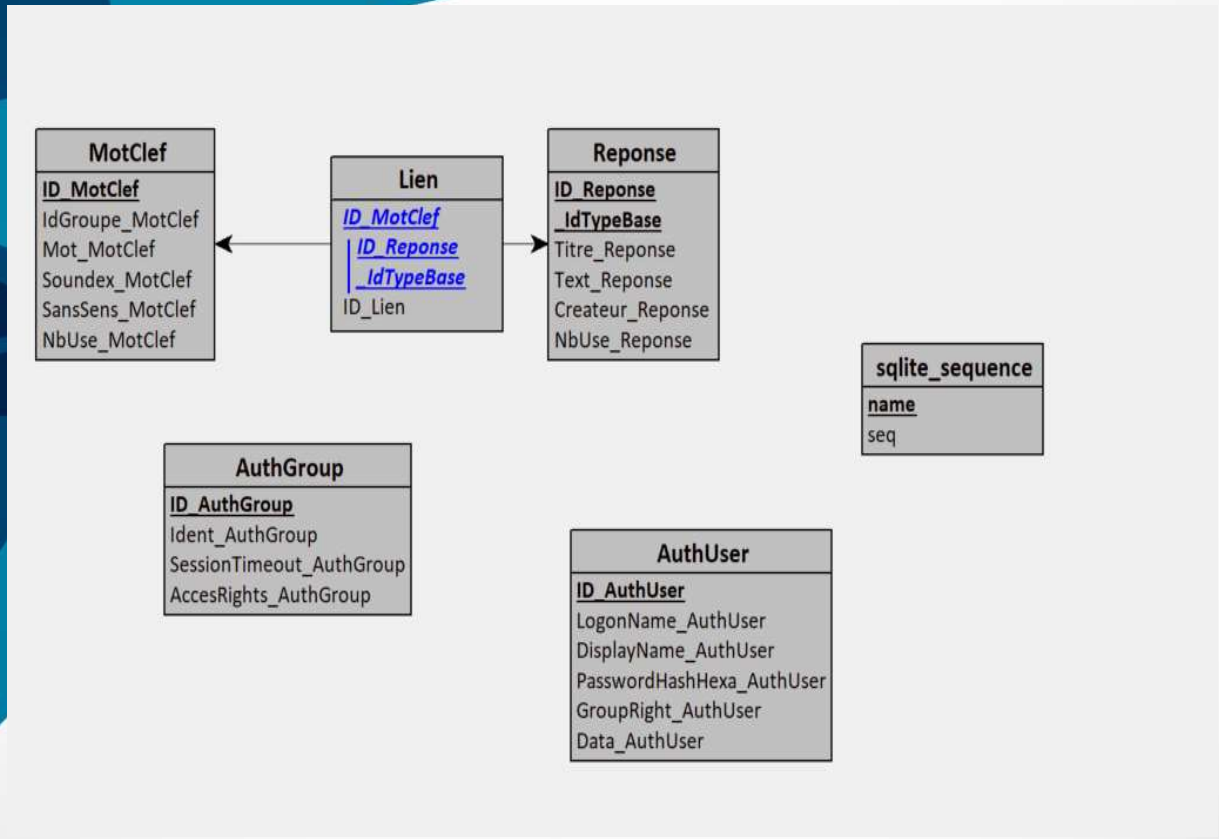
Déclaration des variables dans le fichier assistance-pop-up.component.ts

```
129  // Déclaration de la classe TSQLReponses -> Tableau de TSQLReponse
130  export type TSQLReponses = [TSQLReponse];
131
132  // Déclaration de la classe TSQLReponse
     Mathys, il y a 17 heures | 2 authors (Nicolas and others)
133  export class TSQLReponse {
134    ID:number = 0;
135    Titre: string = '';
136    Text: string = '';
137    IdTypeBase: number = 0;
138    NbUse: number = 0;
139    Createur: string = '';
140  };
```

→ Déclaration des classes TSQLReponses et TSQLReponse dans le fichier TS

Schématisation BDD (Base De Données)

- MLD graphiquement :



Schématisation BDD (Base De Données)

- MCD textuelle :

→ MCD textuelle

MotClef = (ID_MotClef INTEGER, IdGroupe_MotClef INTEGER, Mot_MotClef TEXT, Soundex_MotClef TEXT, SansSens_MotClef INTEGER, NbUse_MotClef INTEGER);

Reponse = (ID_Reponse INTEGER, _IdTypeBase INTEGER, Titre_Reponse TEXT, Text_Reponse TEXT, Createur_Reponse TEXT, NbUse_Reponse INTEGER);

AuthGroup = (ID_AuthGroup INTEGER, Ident_AuthGroup TEXT, SessionTimeout_AuthGroup INTEGER, AccesRights_AuthGroup TEXT);

AuthUser = (ID_AuthUser INTEGER, LogonName_AuthUser TEXT, DisplayName_AuthUser TEXT, PasswordHashHexa_AuthUser TEXT, GroupRight_AuthUser INTEGER, Data_AuthUser TEXT);

sqlite_sequence = (name INTEGER, seq INTEGER);

Lien = (#ID_MotClef, #(ID_Reponse, _IdTypeBase), ID_Lien INTEGER);

Schématisation BDD (Base De Données)

- Création BDD via l'ORM mORMot :

```
TSQLReponse = class(TSQLRecord)
private
  fCreateur: RawUTF8;
  fIDTypeBse: TID;
  fNbUse: Integer;
  fText: RawUTF8;
  fTitre: RawUTF8;
 published
  property Titre: RawUTF8 read fTitre write fTitre;
  property Text: RawUTF8 read fText write fText;
  property IDTypeBse:TID read fIDTypeBse write fIDTypeBse;
  property NbUse: Integer read fNbUse write fNbUse;
  property Createur: RawUTF8 read fCreateur write fCreateur;
 public
  class function AddOrUpdate(Rest:TSQLRest; var Json:RawUTF8):Boolean;
end;
```

```
TSQLLien = class(TSQLRecord)
private
  fIDMotClef: TSQLMotClef;
  fIDReponse: TSQLReponse;
 published
  property IDReponse:TSQLReponse read fIDReponse write fIDReponse;
  property IDMotClef: TSQLMotClef read fIDMotClef write fIDMotClef;
 public
  class function Ajoute(Rest:TSQLRest; IDRep,IDMot:TID):TID;
end;
```

```
TSQLMotClef = class(TSQLRecord)
private
  fIDGroupe: TID;
  fMot: RawUTF8;
  fNbUse: Integer;
  fSansSens: Boolean;
  fSoundex: RawUTF8;
 published
  property IDGroupe:TID read fIDGroupe write fIDGroupe;
  property Mot: RawUTF8 read fMot write fMot;
  property Soundex: RawUTF8 read fSoundex write fSoundex;
  property NbUse: Integer read fNbUse write fNbUse;
  property SansSens:Boolean read fSansSens write fSansSens;
 public
  class function Ajoute(Rest:TSQLRest; aMot:String):TID;
end;
```

Mise en situation

Assistance

Comment éditer mon recu fiscal ?

Titre

754111 - Offrandes affectées nominatives (usage interne à la paroisse);

754123 - Offrandes occasionnelles nominatives (après cérémonies, etc....);

754121 - Offrandes régulières nominatives

771821 - Souscriptions et offrandes affectées exceptionnelles - nominatives

Remboursement d'un emprunt

Edition d'un reçu fiscal en cours d'année

Pour enregistrer les annuités remboursement de prêt :
Aller dans l'écran "Dépense"
Sur le plan officiel, vous choisissez le compte 16 correspondant. Si ce compte n'est pas visible dans la liste déroulante faites un clic droit dans cette zone et cliquez sur l'option "Opération particulière"
Sur le plan interne vous choisissez le compte 115 ou 115-bis en fonction de la nature du prêt (mobilier ou immobilier) et en trésorerie votre compte banque