# Interface Graphique

1. Quelques points qu’on a abordés :
   1. Design : Le design de l'interface graphique doit être assorti avec le design du robot. Du fait qu’on ne connait pas comment va être le design de ce dernier (ça ne fait pas partie des tâches qu'on a à faire cette année), on a décidé de créer un design en se basant sur les couleurs du logo de l'ENSTA et on laisse la possibilité aux groupes des années prochaines de le modifier s'il le veut.
   2. Transfert de données à partir de l’interface vers ROS : création d'un fichier text à partir du code java dans lequel se trouve les informations nécessaires.
   3. On a eu recours à quelques modèles gratuits pour accélérer le processus de développement du site web. Voici quelques références utilisées :
      1. https://freshdesignweb.com/css-login-form-templates/
      2. https://codepen.io/motorlatitude/pen/JFkro
      3. https://codepen.io/cassandraPaige/pen/MWYeqwZ
      4. <https://freefrontend.com/css-page-transitions/>
2. Projet :
   1. Technologies utilisées :
   2. Apache Maven : outil de gestion et d’automatisation de production des projets logiciels Java en général et Java EE en particulier.
   3. Apache Tomcat : conteneur web de servlets et JSP.
   4. JAVA EE
   5. IntelliJ community edition : environnement de développement
   6. Instructions pour utiliser le site :
   7. Commencer par télécharger et installer Maven 3 à partir du site officiel.
   8. Récupérer les fichiers du projet à partir du github.
   9. Installer le logiciel IntelliJ community edition et ouvrir le projet.
   10. Pour initialiser votre base des données, taper cette commande dans le terminal : mvn clean install exec:java . vous devez exécuter cette commande à chaque fois que vous ouvrez le projet.
   11. Pour lancer votre serveur Tomcat, exécutez cette commande dans le terminal : mvn clean install tomcat7:run
   12. Ouvrir votre navigateur et taper localhost:8080/InterfaceGraphique/index.html pour que accéder à la page d’accueil du site web.
   13. Décomposition du projet :
   14. La base des données : on a décider d’utiliser deux tables seulement, un pour modéliser les personnels et l’autre pours les salles. Si vous jugez nécessaire on vous indiquera comment vous pouvez ajouter d’autres tables.
       1. Table Personnel :
          1. Id : clé primaire ; auto incrémenté
          2. Nom : varchar
          3. Prénom : varchar
          4. salleId : clé étranger vers l’id de la salle dans laquelle on peut trouver ce personnel. C’est en général un bureau.
       2. Table Salle :
          1. Id : clé primaire ; varchar(5)
          2. Nom : varchar
          3. Etage : int
          4. X : int
          5. Y : int

Les entier x et y d’une salle sont les coordonnées de la salle dans la mappe utilisée par l’algorithme de recherche de plus court chemin.

* 1. Le paquet com.ensta.interfacegraphique :
     1. Le modèle (model) : ces sont les classes qui modélisent les tables de la base des données dans notre projet JAVA EE
     2. DAO (Data Access Object) : on trouve dans ce paquet les classes qui permettent d’accéder aux données stockées dans la base des données.
     3. Persistence : connexion à la base des données
     4. Utils : contient la classe FillDatabase qui permet de créer la base des données si elle ne l’est pas encore et d’insérer des instances dedans. Dans l’état actuel du site, c’est le seul endroit à partir duquel on peut modifier notre base des données. C’est pourquoi on propose au groupe de l’année prochaine de développer une page d’administration.
     5. Service :
     6. Servlet :
  2. PersonnelDao :
     1. Utilisation du design pattern Singleton dont l’objectif est de restreindre l’instanciation d’une classe à un seul objet. <https://fr.wikipedia.org/wiki/Singleton_(patron_de_conception)>
     2. getlist() : retourne une liste de tous les personnels
     3. getByString(string) : retourne une liste de personnel dont le nom ou le prénom ou la salle au quelle il est lié contient la chaine de caractères passée en arguments.
     4. getById(int) : retourne le personnel dont l’id est égale à l’entier passé en arguments.
     5. create(nom,prenom,salleId) : permet d’insérer un nouveau personnel au table personnel est retourner son identifiant.
     6. update(personnel) : mettre à jours les informations du personnel passé en paramètres.
     7. delete(id) : supprime le personnel dont l’id est égale au entier passé en arguments.

Dans l’état actuel du site nous avons besoin seulement de la fonction getByString mais les autres méthodes vous seront utiles si vous décidez de développer une page d’administration avec laquelle on peut gérer la base des données.

* 1. SalleDao : on a quasiment les mêmes méthodes de la classe PersonnelDao, sauf pour la méthode getByNameorId(string) qui retourne la liste des salles dont l’identifiant ou le nom contient la chaine passée en paramètres.
  2. DashboardServlet : c’est la servlet liée à la page du résultat de recherche (fichier dashboard.jsp). Elle traite la requête de recherche en déterminant le critère du recherche (la chaine écrite dans la barre de recherche) et fait appel au PersonnelService et SalleService pour récupérer les listes des personnels et des salles et les passer vers la page dashboard.jsp.
  3. FollowMeServlet : cette servlet est appelée lorsqu’on demande au robot de nous guider vers notre destination. Elle traite la requête correspondante en déterminant les coordonnées x et y de la position pour les passer vers le script ROS (ça n’a pas été implémenté). Elle nous redirige vers la page followme.jsp qui affiche « FOLLOW ME » sur l’écran puis elle attend que le robot arrive à sa destination (vous pouvez faire ceci en utilisant une boucle qui attend à ce qu’un fichier soit disponible dans un emplacement bien déterminé et générer ce fichier avec le code de déplacement)
  4. WEB-INF
     1. View : placer vos fichier .jsp dans ce dossier
     2. Web.xml : permet de faire le mapping entre les URL et les servlets. Pour ce faire, commencez par définir votre servlet dans une balise <servlet> (vous trouverez des exemples dans le fichier du projet) puis définissez le servlet mapping avec la balise <servlet-mapping>. En fin, dans votre classe servlet correspondant ajoutez @WebServlet(‘/servlet-name’) (sevlet-name est à spécifier dans le fichier web.xml)
     3. Index.html : c’est la page d’accueil du site
  5. Assets
     1. Style.css : le fichier css utilisé par les fichiers html et .jsp.
     2. Js : contient les scripts javascript utilisés dans les pages html et .jsp
        1. CheckInput.js : ce script permet d’assurer que le bouton de recherche de la page d’accueil ne soit activé que lorsque l’utilisateur à écrit quelque chose dans le champ d’entrée.
        2. ShowHidePanel.js : Contient 3 fonctions :
           1. ShowHide(string,int,int) : permet d’afficher ou cacher le panneau de la page de recherche qui est activé lorsqu’on clique sur une ligne du résultat de recherche.
           2. HomePage() : permet de retourner vers la page d’accueil du site en cliquant sur le titre en haut à gauche de la page. On propose aussi que vous ajoutez une icône « home » qui a le même rôle.
           3. Redirect(x,y) : rediriger vers la page followme.jsp en passant les coordonnées x et y à la requête.
  6. Fichier pom.xml (Project Object Model): contient les informations nécessaires à Maven pour traiter le projet (nom du projet, numéro de version, bibliothèques nécessaires à la compilation..etc).
  7. Méthode de test :

La classe ImplTest est utilisée pour tester les différentes classes du projet. Initialiser une instance de la classe que vous souhaitez tester et l’utiliser et voir si tous fonctionnent comme prévu.