Générateur de formulaire GUI

☑ Rapport de projet tutoré S3

Equipe	:	Infotik
Date	:	15/01/2018
Annee	:	2017-2018
Projet	:	Projet tutoré S3
Enseignants	:	M. Legrix, M. Le Pivert, Mme. Boukachour, M. Duflo

Générateur de formulaire GUI

Rapport de projet tutoré S3

Table des matières

Introduction

Durant le semestre trois, il nous a été proposé dans le cadre du projet tutoré de répondre à une offre de travail proposée par l'association Prom'Info, incarnée par nos enseignants. Dans cette offre, la société nous demandait de participer à trois entretiens pendant lesquelles nous avons pu capturer leurs besoins. Avec ces informations, nous avons pu leur présenter au cours d'une soutenance orale le cahier des charges que nous avons rédigé sur leur application, accompagné d'une offre commerciale.

Il s'agit d'un générateur de formulaires généré à partir d'un fichier XML simplifié à l'extrême écrit par l'étudiant, destiné aux étudiants de fin de S1 n'ayant aucune réelle expérience en programmation. Le but est de leur permettre de rendre plus visuel leur façon de programmer, et donc de simplifier leurs affectations de variables.

Pour garantir l'uniformité des solutions pour la correction, les professeurs nous ont par la suite fait part de leur propre version du cahier des charges, avant de nous demander de développer l'application.

Le but de tout ceci était de nous professionnaliser au maximum avant la fin d'année, en nous mettant dans une situation très proche de celles que l'on rencontrera en entreprise une fois avec une emploi, et même peut-être plus tôt lors de notre stage.

I. Une démarche bien huilée

1. Les entretiens : commencer un projet dans les règles en répondant à une offre

a. Premier entretien : capter le besoin

Le but du premier entretien était le besoin du client (Prom'Info). Pour ce faire, nous (notre équipe : Infotik), avions préparées une série de questions à poser à nos clients, parmi lesquelles par exemple :

Liste des questions (non exhaustive)
« Une idée de la structure générale de l'application ? »
« Avez-vous des préférences concernant l'aspect de l'IHM ? »
« Doit-on pouvoir gérer plusieurs écrans à la fois ? »
« Les utilisateurs ont-ils à associer une classe ? »
« Quels doivent être les contrôles à gérer dans le formulaire ? »

Liste des questions posées lors du premier

Les deux professeurs rencontrés lors de ce premier entretien, M. Le Pivert et M. Duflo, se présentaient comme des non informaticiens : ils ne pouvaient donc pas répondre à des questions techniques. Ceux-ci étaient donc là pour nous guider et nous aider à préciser leurs attentes.

b. Deuxième entretien : cerner les spécificités techniques

Lors de cet entretien, nous avons rencontré la deuxième équipe, plus technique, de Prom'Info. Elle était incarnée par deux nouveaux professeurs, M. Legrix et Mme. Boukachour. Ceux-ci se présentaient comme les informaticiens de l'association et étaient là pour répondre à toutes nos questions techniques comme : « Avez-vous une préférence dans l'utilisation d'un langage en particulier ? ».

Cela nous a alors permis de procéder à l'élaboration d'un cahier des charges ainsi qu'à réaliser une proposition commerciale. Ceux-ci devaient être prêts à être présentés lors de la troisième réunion.

c. Troisième entretien : proposer un cahier des charges

Lors du 3ème entretien, nous avons de nouveau rencontré M. Le Pivert afin de lui présenter notre cahier des charges. C'est aussi durant cet entretien que nous avons présenté notre proposition commerciale qui s'élevait à hauteur de 8 316 €.

Grâce à cet entretien, M. Le Pivert nous a apporté des précisions supplémentaires sur le contenu du cahier des charges, en nous disant quoi rajouter avant la soutenance. Par exemple, il nous a été reproché de ne pas avoir rangé les fonctionnalités par importance.

2. Soutenance de cahier des charges

Pour cette soutenance, nous étions jugés par un panel de professeurs représentant l'association Prom'Info. Chaque équipe présentait leur cahier des charges devant les deux autres groupes et le jury. A la fin le jury portait des commentaires sur nos présentations et nous posait des questions sur nos choix par exemple.

Mais ce cahier des charges n'était là que pour permettre aux professeurs de juger de nos compétences à en élaborer un car un nouveau (créé par les professeurs) ainsi que des spécificités techniques ont été mise en ligne afin de nous permettre de réaliser le projet.

3. Après la théorie, la pratique : passage à la conception de l'application

Avant de coder, nous avons réalisé une ébauche de diagramme UML pour avoir une idée des interactions entre les différentes classes que nous allions créer, et comment nous allions répartir les fonctionnalités.

Nous avons articulé notre programme autour d'une simple classe contrôleur appelée FormController, qui contient les ensembles dans lesquels sont rangés les différentes valeurs rentrées dans le formulaire par l'utilisateur, accessible à l'aide de méthodes telles que **getBoolean**, **getArray** ou encore **getInt**, avec une méthode pour chaque type de base. Pour rappel, les cinq types de base sont : l'integer, le double, la chaîne, le booléen et le caractère.

Voici un zoom sur la partie centrale du diagramme UML de l'application, gérant et créant la fenêtre :

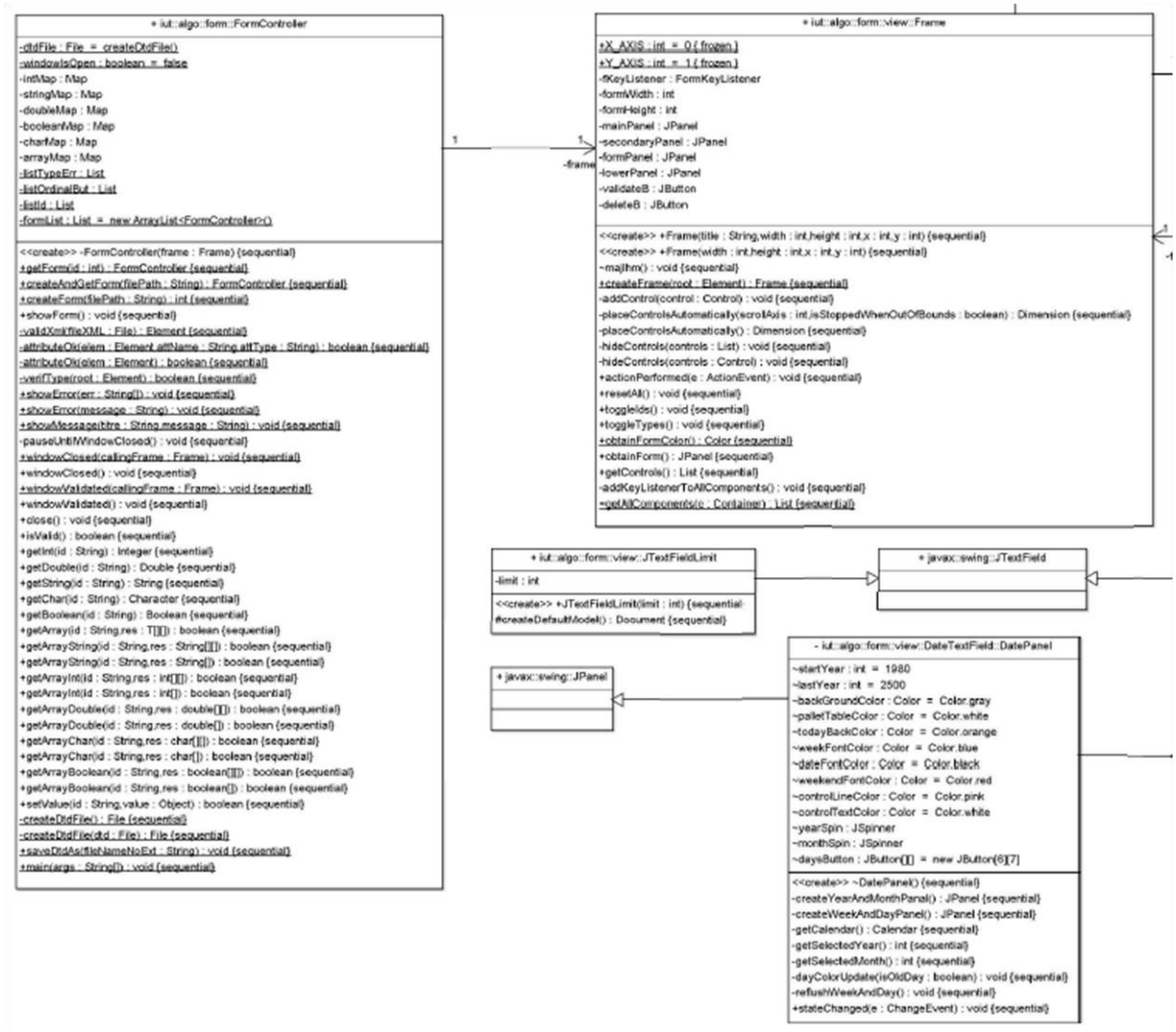


Diagramme UML du contrôleur et de la fenêtre

4. Nos outils, une équilibre entre choix communs et personnels

a. Les éditeurs de texte : des choix variés

Dans notre équipe, chaque membre utilise un éditeur de texte de son choix. Adam, lui, décide d'utiliser **Notepad++,** simple d'utilisation et plein de fonctionnalités, car il n'a jamais souhaité en utiliser d'autres, satisfait par celui-ci.

Florent, quant à lui, utilise **Sublime Text** car ce logiciel lui a été introduit pendant sa première année de licence à l'université du Havre, et qu'il le satisfait, étant un bon compromis entre IDE et éditeur de texte et disponible de base avec un thème sombre.

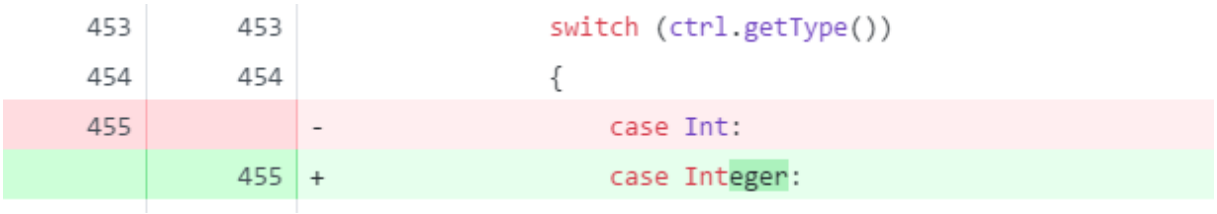
Enfin, les autres membres du groupe utilisent **Atom**, car celui-ci se présente comme un IDE adapté à tous types de langages. Il est aussi très puissant et intègre directement l'add-on GitHub, simplifiant le partage de données.

b. Le partage de travaux et de données par GitHub

GitHub est un outil permettant de travailler en équipe sur un projet informatique. En effet, celui-ci permet d'apporter des modifications aux mêmes fichiers en même temps et d'avoir un suivi de projet très poussé.

Ce suivi passe par ce qu'on appelle des « commits », qui permettent d'accompagner les modifications apportées d'un commentaire et d'un titre, pour en signifier le contenu à ses collègues. Chaque commit contient aussi une visualisation des modifications, avec l'ancienne version sur un fond rouge, et la nouvelle sur un fond vert. Ce qui n'a pas été modifié est sur fond blanc.

Ci-dessous, une petite image pour illustrer :



Exemple de commit GitHub

Sa notoriété et son côté pratique ont donc tout naturellement porté notre choix vers ce logiciel. De plus, nous l'avions déjà utilisé lors du projet précédent.

c. ArgoUML, un logiciel pour faire de l'UML

Il s'agit du seul logiciel suffisamment abouti que nous avons trouvé qui permet de générer des diagrammes UML à partir de fichiers java. Nous nous en sommes servis pour créer le diagramme de classe présent dans la documentation programmeur. Ce logiciel n'a pas été utilisé par choix, mais par nécessité, et nous avons malheureusement été ralentis par son instabilité.

II. Un apprentissage sur le long terme

1. Un projet après de nombreux autres : une expérience qui en découle

En effet, au cours de tous les projets précédents que réalisés à l'IUT, nous avons pu, après avoir tâtonné, nous créer quelques outils réutilisables qui accélèrent grandement notre temps de développement.

Par exemple, nous avons écrit des fichiers « .bat », pour Windows, et « .sh », pour Linux, pour compiler et exécuter nos fichiers, et tout ça en ne lançant qu'un fichier. Nous avons aussi un programme capable d'écrire dans un fichier texte le nom de tous les fichiers sources et utilisé par le « .bat » de l'arborescence à compiler, pour ne pas avoir à les ajouter à la main.

2. Un suivi constant

Pendant tout le projet, l'un d'entre nous, principalement Adam et Martin, tenaient à jour un fichier contenant la liste des choses à faire, accompagnées de leur importance, et un « Lisez-moi » listant les différentes fonctionnalités.

Ici, on peut voir un extrait de la « To do » liste :

```
# Voici ce qu'il faut faire par ordre d'importance
### Legende
- [ ] Je ne suis pas fini
- [x] Je suis fini mais pas testé
- [x] **Je suis fini et testé**

## Primordial
#### [x] Paramétrage du formulaire
- [x] **Le formulaire devra être paramétré dans un fichier texte**
- [x] **Les 5 types de bases devront être gérés :**
    - [x] **int**
    - [x] **boolean**
    - [x] **String**
    - [x] **char**
    - [x] **boolean**
- [x] **L'utilisateur pourra définir plusieurs formulaires et utiliser plusieurs formulaires dans un même programme.**

#### [x] Les différents Contrôles à gérer.
Les différents Contrôles devront posséder :
- [x] **une étiquette associée**
- [x] **un  identifiant numérique, indépendant de l'ordre dans lequel les contrôles sont définis**
- [x] **un positionnement Sans valeur x y les différents Controles seront placées les uns en dessous des autres dans l'ordre de leur identifiant**
- [x] **la définition de la largeur (qui par défaut sera de 150)**
- [x] **la hauteur sera fixe, et correspondra à une hauteur standard**
- [x] **L'association à un des 5 types de base**

#### [x] Les différents Contrôles à gérer sont
- [x] **La zone de texte**
- [x] **Une case à cocher**
```

Extrait de la liste des choses à faire (sous SublimeText)

Ce suivi nous a permis de ne jamais être perdus et de toujours savoir quoi faire après avoir implémenté une fonctionnalité.

III. Un regard critique sur nos actions

1. Des difficultés peu présentes et peu impactantes

Au cours du projet, nous n'avons pas vraiment rencontré de problèmes liés à la programmation ou à notre organisation, mais plus du côté des outils qui nous sont mis à disposition.

a. La connexion à l'IUT

Premièrement, nous avons eu du mal à nous connecter au réseau wifi de l'iut. En effet, avec l'ensemble des personnes en projet connecté au wifi de l'iut, celui-ci n'était plus assez puissant pour garantir un accès constant à Internet. Or, pour pouvoir travailler en réseau avec **GitHub**, une connexion stable est nécessaire, ce qui nous a obligés à utiliser une connexion 4G personnelle tout au long de la semaine.

b. Une salle plus calme

Il aurait été agréable d'avoir une salle à nous pour limiter le bruit et pouvoir mieux réfléchir, bien que nous ayons conscience que ce soit inconcevable, le nombre de salles étant limité.

c. Un logiciel instable : ArgoUML

Martin a rencontré d'énormes difficultés avec le logiciel d'UML que nous avons utilisé, **ArgoUML**. Ce logiciel n'étant pas vraiment stable, son utilisation devenait compliquée. Il serait en effet intéressant d'arriver à trouver un autre logiciel plus stable pour de futurs projets.

2. Echouer pour mieux réussir

Si nous avons pu être si efficaces pendant ce projet, c'est parce que nous avons appris de nos erreurs suite aux projets précédents. En effet, au cours des projets précédents, nous avons développés de nombreux outils qui nous ont servi et nous serviront encore par la suite, comme dit plus haut.

Pour la suite, nous avons l'intention de développer un programme rajoutant un modèle à compléter des commentaires Javadoc, et un autre formatant le code selon un modèle prédéfini.

3. Quelques éléments à revoir

a. Pousser l'organisation de l'arborescence plus loin

Dans le dossier racine de notre projet, nous avons un dossier « src » contenant les fichiers sources, les *.java*, et un dossier « bin », contenant les fichiers compilés, les *.class*. Mais pour le reste, les fichiers texte par exemple, tout est placé et mélangé dans le dossier racine. Pour les projets suivants, il serait donc utile de séparer encore plus les fichiers, pour les retrouver plus facilement et gagner du temps.

b. Rendre nos tests plus professionnels

Pour faire nos tests, nous avons utilisé un unique fichier « Demo.java » que nous modifiions tous sans cesse. En plus de créer des difficultés à GitHub, qui ne savait plus quelle modifications garder, cela nous aurait permis de conserver nos tests et de nous simplifier la tâche à la fin du projet, lorsqu'il a fallu vérifier si tout fonctionnait.

Il aurait été judicieux de créer des tests **JUnit** ou équivalents pour les conserver et pouvoir tester de nouveau après les potentielles modifications du programme.

Conclusion

Au final, nous sommes satisfaits de la façon dont a été mené le projet, avec une organisation qui s'améliore au fur et à mesure. Bien que certains points peuvent toujours être améliorés, nous avons pu perdre le moins de temps possible en utilisant des outils développés pour les projets antérieurs. De plus, nous avons d'ores et déjà prévu d'en développer de nouveaux pour pallier aux problèmes rencontrées pour la réalisation du formulaire.

Le principal point à améliorer pour la prochaine fois est notre façon de vérifier si notre code fonctionne, en utilisant des tests unitaires mieux encadrés et conservables. Nous espérons ainsi pouvoir être encore plus efficaces et réaliser la maintenance de notre programme avec le moins de difficultés possibles.

Table des illustrations