

Caravana FILS

-Système de gestion des visites au lycées-

Projet réalisé par DOVLETE Robert & CONSTANTIN Daniela

Avec une coordination minutieuse de la part de

Prof. Maria-Iuliana Dascălu et Assist. Iulia-Cristina Stănică

A close up of a sign

Description automatically generated

1.Brève présentation

-Notre application a pour objectif principal la simplification et l'intégration de tous les aspects concernant les visites d'étudiants et d'enseignants du FILS dans les lycées, étant également un outil pour les représentants des lycées qui souhaitent prendre rendez-vous pour une éventuelle visite.

La raison du développement de l'application est le processus long et incertain qui existe maintenant pour les visites.

De plus, l'application est également disponible pour les étudiants, ce qui les motive à faire partie de FILS dans tous les aspects.

# acteurs

* étudiants
* Professeurs de FILS
* Administrateur d'application (doyen)
* Représentant du lycée (la personne qui gère les visites)
* Personnel technique responsable de l'application

# Fonctionnalités de l'application

* créer un compte
* Connectez-in / Log-out
* Planifier une visite
* Téléchargez des photos pendant les présentations
* Possibilité pour le doyen de nommer les professeurs et étudiants présents aux présentations
* Mise à jour du profil (nom, photo de profil, etc.)

# règles métier (business rules)

* Un nombre minimum de caractères et des conditions spéciales pour utiliser un mot de passe (un nombre, un caractère spécial, etc.) BR1
* Deux utilisateurs de l'application ne peuvent pas utiliser le même utilisateur. BR2
* Seul le doyen peut désigner les enseignants et étudiants présents lors des futures presentations. BR3
* Seuls le doyen et l'administrateur des présentations de l'école respective peuvent accepter/annuler les demandes de presentation. BR4
* Seuls les étudiants, les enseignants et l'administrateur des présentations de l'école respective peuvent télécharger des photos prises lors des presentations. BR5
* Seuls 2 étudiants / présentation + 1 enseignant maximum peuvent être acceptés.BR8
* L'heure à laquelle la visite est prévue doit être jusqu'à 17 h 00, uniquement les jours ouvrables. BR6
* Chaque utilisateur doit sélectionner le bon type de compte (par exemple, un étudiant ne peut pas sélectionner un compte de type "doyen"). BR7
* Il ne peut pas y avoir deux visites simultanément dans le même lycée, il doit y avoir 1 heure entre elles. BR9

# Matrice RASIC

A picture containing cabinet

Description automatically generated

A screenshot of a social media post

Description automatically generated

2. Diagrammes UML

# Cas d’utilisation

A close up of a map

Description automatically generated

|  |  |
| --- | --- |
| Acteurs | * Etudiants * Professeurs de FILS * Administrateur d'application (doyen) * Représentant du lycée   *Tous les types d'utilisateurs qui peuvent créer un compte (sauf les administrateurs d'application)* |
| Règles métiers | BR1, BR2, BR7  *Chaque BR est intuitivement représenté sur le diagramme* |
| Sommaire | Nous avons décrit la situation de la création d'un nouveau compte.  *Notre diagramme représente la vue d'ensemble de l'application, de la création du compte à la confirmation de la visite.*  *Nous avons choisi de faire un seul diagramme pour une meilleure compréhension des fonctionnalités de l'application et d'avoir une vision plus claire de celui-ci pendant que nous le développons.*  *Cette méthode est meilleure pour nous aussi pour l'étape de révision.* |
| Précondition(s) | Chaque nouvel utilisateur a besoin d'un numéro de téléphone, d'un e-mail et d'une photo de profil pour créer un nouveau compte sur l'application.(en respectant les règles métier spécifiques). |
| Post-condition(s) | Un nouvel utilisateur est ajouté à la base de données. |
| Pas | 1. L'utilisateur sélectionne l'option "Créer un nouveau compte"  2. L'application affiche la page concernant la création d'un nouveau compte.  3. L'utilisateur doit remplir le formulaire avec un nom, un numéro de téléphone et un e-mail et doit sélectionner son type de compte.  4. Le formulaire est vérifié et approuvé par le système  5. L'utilisateur est obligé par le système de choisir un mot de passe  6. L'utilisateur introduit un mot de passe.  7. Le système valide le mot de passe  8. L'utilisateur reçoit un message de confirmation.  9. L'utilisateur est ajouté à la base de données. |
| Exceptions | 2.1. Il y a un échec de l'application lorsqu'elle essaie de se connecter à Internet.  3.1. L'e-mail/ numéro de téléphone n'est pas valide ou est déjà utilisé. (BR2)  4.1. L'utilisateur ne fait pas partie de FILS.  4.2 . Le type de compte n'est pas le bon (par exemple, un student n'a pas pu sélectionner un type ''doyen'') (BR7)  4.3. Impossible de vérifier les informations d'identification de l'utilisateur. Mauvaise connexion avec le BD.  7.1. Le mot de passe ne respecte pas les conditions. (BR1)  8.1./9.1. Il est impossible de finaliser la création du compte en raison d'une mauvaise connexion avec le BD. |

# Diagrammes de séquence

A screenshot of a social media post

Description automatically generated1) Ce diagramme présente la façon dont un compte est créé.

A screenshot of a social media post

Description automatically generated2) Ce diagramme présente le mode d'établissement d'une visite par le doyen.

3) Ce diagramme présente la façon dont les utilisateurs autorisés peuvent ajouter des photos ou des vidéos de la visite.

A screenshot of a social media post

Description automatically generated

# Diagrammes d’activités

1. A screenshot of text

   Description automatically generatedCe diagramme présente la façon dont un compte est créé.

A screenshot of a cell phone

Description automatically generated2) Ce diagramme présente la façon dont les utilisateurs autorisés peuvent ajouter des photos ou des vidéos de la visite.

3) Ce diagramme présente le mode d'établissement d'une visite par le doyen.

A close up of a map

Description automatically generated

# Diagramme de paquets

A picture containing screenshot

Description automatically generated

3.Description du modèle architecturale

# Design patterns

1. DAO – Data Access Object

Ce modèle est utilisé pour séparer les données de bas niveau des services commerciaux – «business» de haut niveau. Ainsi on va délimiter clairement les classes qui s’occupent avec les opérations sur la base des donnes (opérations CRUD ou autres plus spécifiques à notre projet) en utilisant un concept appelle « séparation de la logique ». Le service reste complètement dans l'ignorance de la façon dont les opérations de bas niveau sont effectuées pour accéder à la base de données.

Components :

· Le modèle – chez nous la classe « User » - qui est transféré d'une couche à l'autre

· L’interface – « Dao » tout simplement - qui offre une conception flexible

· L'implémentation d'interface – « UserDao » - qui est une implémentation concrète de la logique de persistance

2. SINGLETON

Le concept lors du choix d'un singleton est qu'il garantit qu'une seule instance sera créée. Il existe une instance privée d'un singleton et un constructeur privé. C'est parce que nous voulons que le singleton lui-même appelle le constructeur et personne d'autre. Il résout un problème très bien défini où nous ne voulons avoir qu'une seule instance de cet objet à l'intérieur de notre application.

Pour notre projet, nous allons créer un singleton pour gérer l'accès à qui peut créer des connexions à notre base de données et où ils peuvent les créer. Cela garantit que d'autres objets ne peuvent pas initialiser des connexions inutiles qui ralentiraient les performances globales du système en raison de multiples instanciations. En moins de mots: motif singleton => garantie d'une seule instance de notre base de données => amélioration des performances.

3. BUILDER

Lorsque nous envisageons un motif builder, nous souhaitons nous concentrer sur la complexité de la construction d'un objet. Par complexe, nous parlons spécifiquement de beaucoup d'arguments pour un constructeur, ou de beaucoup de setters. Par conséquent, ce modèle est idéal pour gérer la construction d'objets qui peuvent contenir beaucoup de paramètres et nous voulons rendre l'objet immuable une fois que nous avons fini de le construire.

Le modèle de générateur résout un problème très courant dans la programmation orientée objet et qui détermine quel constructeur utiliser. Souvent, les gens créent plusieurs constructeurs et cela peut devenir difficile à gérer. La création de plusieurs constructeurs avec chaque variation de paramètre est appelée constructeur télescopique. Le modèle de builder nous a aidés à surmonter cette difficulté. Le builder travaille à l'unisson avec tous les constructeurs et appelle celui qui convient en fonction de son état. Par conséquent, il ne nie pas les constructeurs. Cependant, il annule le besoin de setters exposés.

Dans notre projet, nous allons utiliser le modèle Builder pour la classe « Visit ». En raison de la multitude d'arguments présents dans la classe (lycée, date, professeur FILS assistant à la visite, ainsi que les étudiants FILS et le représentant du lycée) nous avons découvert qu'il est difficile de gérer tous les constructeurs, avec toute la variation des paramètres. La tâche devient encore plus lourde car les objets de différentes classes ajoutent des instances à l'objet Visit: le doyen décide quel professeur et quels étudiants du FILS seront présents, le responsable du lycée s'ajoute lui-même dans les détails de la visite et confirme la date, etc.

# Diagramme de classes

A screenshot of a computer

Description automatically generated

# Diagramme de la base de données

A screenshot of a cell phone

Description automatically generated