Projet de Module Systèmes Répartis & Distribués

Contexte et Objectif du Projet

Le but de ce projet est d'implémenter un système distribué en utilisant des technologies de communication à distance telles que **RPC (Remote Procedure Call)**, **RMI (Remote Method Invocation)**, ou **CORBA (Common Object Request Broker Architecture)**. Ce système distribuera les différentes parties de l'application sur plusieurs machines ou serveurs et permettra la communication entre eux pour accomplir des tâches spécifiques.

Chaque équipe sera composée de **4 étudiants**, et chaque étudiant devra être responsable d'une ou plusieurs parties du projet en fonction de ses compétences. L'utilisation de **GitHub** est obligatoire pour gérer le code, suivre les versions et faciliter la collaboration entre les membres de l'équipe.

Les projets devront aussi intégrer une **interface web** permettant aux utilisateurs d'interagir avec le système distribué à travers un navigateur.

Technologies et Outils Requis

- **Backend (Serveur RPC, RMI, CORBA)** :
  - **Python** (pour RPC), **Java** (pour RMI) & **C++ ou Java** (pour CORBA)
- **Frontend (Interface Web)** :
  - **Python/Django** (pour l'interface web en Python)
  - **Java** (Java EE ou Spring pour le backend, JSP/Servlet pour l'interface)
  - **HTML/CSS/JavaScript** (pour l'interface front-end de base)
- **Gestion de version** : GitHub pour le contrôle de version et la collaboration.

Plan du Projet

A. Choix du Projet et Définition des Rôles

- Formez une équipe de 4 étudiants.
- Choisissez un projet parmi ceux proposés (voir document partagé).
- Assignez les rôles : Backend, Frontend, Intégration, et Gestion des tests/qualité.

B. Phase de Conception et Planification

Objectifs du projet : Clarifiez les objectifs du projet (par exemple, création d'un système de réservation de salle, ou système de messagerie instantanée).

Architecture distribuée : Concevez l'architecture distribuée, en définissant comment les différentes parties du système communiqueront entre elles via RPC, RMI ou CORBA.

Diagramme de flux de données : Créez un diagramme illustrant comment les données seront échangées entre les clients et les serveurs.

C. Configuration de l'Environnement de Travail

Créer un dépôt **GitHub** pour le projet et initialiser les fichiers nécessaires. Configurer un environnement de développement local (par exemple, installation de Java, Python, Django, serveur CORBA, etc.).

D. Développement

Backend (Serveur)	• Implémenter un serveur RPC, RMI ou CORBA en fonction du projet. • Si vous utilisez RMI, assurez-vous de créer des interfaces et des méthodes distantes accessibles aux clients via la technologie **RMI**. • Assurez-vous que les méthodes distantes soient correctement définies pour les appels clients.
Frontend (Interface Web)	• Développez une interface utilisateur conviviale avec HTML, CSS et JavaScript. • Si vous utilisez **Django**, créez les vues et les templates pour l'interaction utilisateur avec le serveur. • Si vous utilisez **Java EE**, configurez une interface web (JSP/Servlet) pour permettre à l'utilisateur de saisir des données ou interagir avec le backend.
Communication Distribuée	• Assurez-vous que les appels distants sont correctement configurés pour communiquer entre le client et le serveur. • Implémentez la gestion des erreurs liées à la communication à distance, comme la gestion des pannes ou des problèmes de réseau.

E. Tests

Tests unitaires	Rédigez des tests pour chaque composant du système. Par exemple, pour le serveur, testez les appels distants. Pour l'interface web, testez la soumission des formulaires et l'affichage des résultats.
Tests d'intégration	Testez l'intégration entre le frontend et le backend. Vérifiez que les méthodes distantes fonctionnent correctement lorsque le client envoie des requêtes.
Tests de performance	Effectuez des tests pour évaluer la performance du système distribué, en particulier en cas de surcharge de requêtes.

Bonnes pratiques

1. **Planification** : Organisez-vous en définissant des étapes claires avec des délais pour chaque fonctionnalité du projet.
2. **Communication** : Communiquez régulièrement avec les autres membres de votre équipe via GitHub, Slack, ou toute autre plateforme.
3. **Prototypage** : Faites un prototype de l'interface utilisateur avant de commencer le développement pour avoir une idée plus précise de ce que vous voulez réaliser.
4. **Gestion des erreurs** : Prenez en compte les erreurs possibles dans les communications distribuées (par exemple, des échecs de réseau, des appels distants échoués, etc.) et gérez-les proprement.

Rapport du Projet

Le rapport sera structuré en plusieurs sections principales pour décrire le projet et fournir des informations détaillées sur l'architecture, les choix techniques et les diagrammes UML. Voici un guide pour l'organisation du rapport.

1. Introduction

Objectif du projet : Décrire brièvement l'objectif du projet et Préciser les technologies utilisées : RPC, RMI, CORBA, ainsi que l'interface web.

Contexte : Pourquoi un système distribué est nécessaire pour ce projet. Les défis spécifiques auxquels vous vous êtes attaqués (par exemple, gestion des communications entre plusieurs machines, la scalabilité du système, etc.).

2. Architecture Logicielle

2.1 Description générale de l'architecture du système

- **Vue d'ensemble de l'architecture** : Cette section doit expliquer de manière globale comment le système distribué est conçu et comment les différentes parties interagissent entre elles.
- **Types de communication** : Indiquer si le système utilise **RPC**, **RMI**, ou **CORBA**.
- **Rôle de chaque composant**
  - **Serveur** : Le rôle du serveur qui gère les appels distants.
  - **Client** : La partie client qui initie des appels distants.
  - **Base de données** : comment les données sont stockées et partagées.
  - **Interface Web** : Description de l'interface utilisateur et de son interaction avec le backend.

2.2 Déploiement distribué

- Indiquer comment le système est déployé dans un environnement distribué (par exemple, serveurs distincts, utilisation de machines virtuelles, cloud computing, etc.).
- Décrire la topologie du réseau et la gestion des appels distants (comment les clients se connectent au serveur, gestion des sessions, etc.).

3. Diagrammes UML

Les diagrammes UML servent à visualiser la structure du système, l'interaction entre les composants, et les flux d'information. Voici les diagrammes recommandés :

- **Diagramme de cas d'utilisation**
- **Diagramme de classes**
- **Diagramme de séquence**
- **Diagramme de déploiement** (pour représenter l'architecture physique du système et les relations entre les composants matériels (serveurs, clients) et les composants logiciels (applications, services distants) et pour montrer la répartition du système sur différentes machines ou serveurs dans un environnement distribué.

4. Détails Techniques et Implémentation

Choix des technologies : Discuter des avantages et des inconvénients de chaque technologie.

Mise en œuvre du système distribué :

- Décrire en détail l'implémentation des composants distants (serveur, client).
- Expliquer comment les appels distants sont faits et comment la communication est gérée.
- Si nécessaire, inclure des fragments de code et expliquer les étapes clés de l'implémentation.

Gestion des erreurs et des pannes :

- Expliquer comment les erreurs de communication sont gérées, par exemple : gestion des échecs de réseau, des erreurs de sérialisation, et des exceptions dans les appels distants.

5. Conclusion

- Résumer les principaux résultats du projet, ce qui a été accompli et comment le système répond aux exigences du projet.
- Mettre en lumière les défis techniques rencontrés et les leçons tirées.
- Mentionner des pistes d'amélioration ou des extensions futures possibles du système.

7. Références

- Inclure toutes les ressources utilisées pour réaliser le projet : documentation technique, tutoriels, articles de recherche, livres, etc.

Présentation

La présentation doit se concentrer sur le **fonctionnement du système distribué** et mettre en évidence les aspects suivants :

A. **Objectifs et contexte** du projet.

B. **Architecture du système** : Présentation de l'architecture logicielle et physique du système distribué, avec des diagrammes UML pour illustrer.

C. **Communication distribuée** : Expliquez comment les différentes parties du système communiquent entre elles via RPC, RMI, ou CORBA.

D. **Démonstration fonctionnelle** : Montrez l'application en action. Comment les utilisateurs interagissent avec l'interface web et comment les appels distants fonctionnent.

E. **Conclusion** : Résumez ce qui a été accompli et les défis surmontés.

Remarques :

- Tout projet qui ne respecte pas ce descriptif sera rejeté.
- La note du projet représente 60% de la note du module
- Les livrables du projet **doivent être envoyés par email 1 jour (24H) avant la soutenance**
- Les livrables : Rapport (Word+PDF) + Présentation PowerPoint + Projet compressé /lien GitHub
- Les soutenances suivront l'ordre chronologique des projets

26/03/2025	10/04/2025	17/05/2025
La date limite d'inscription	La validation des cahiers de charges	Début des soutenances