Rapport de projet

Gestionnaire de réservations

1. Résumé du projet

Ce projet consiste en le développement d'un gestionnaire de réservations en application console, Écrit en C#, permettant de gérer des ressources, des clients et des réservations. L'objectif est de fournir une solution simple, modulaire et fonctionnelle, reposant sur une architecture orientée services.

But du projet :

Mettre en place un système de réservation console simple et extensible.

Etat actuel :

- ✓ Application console fonctionnelle.
- ✓ Architecture basée sur une couche de services (`RessourceService`, `ServiceReservation`, `ConfigurationService`, `ClientService`).
- ✓ Interface utilisateur pilotée par `MenuPrincipal`.
- ✓ Validation des données assurée par `ConstraintService`.

2. Fonctionnalités principales

L'application offre les fonctionnalités suivantes :

- ✓ *Création et gestion des ressources:* salles et matériels avec leurs caractéristiques.
- ✓ *Gestion des réservations*: enregistrement, suivi et détection des conflits de réservation.
- ✓ *Gestion des clients* : création et gestion des informations clients.
- ✓ *Configuration applicative* : paramètres globaux via `Configuration Service`.
- ✓ Validation des données **:**
- ✓ Validation des adresses email.
- ✓ Validation des numéros de téléphone.
- ✓ Implémentation via `GererContraintes/ConstraintService.cs` à l'aide d'expressions régulières.

3. Architecture technique

3.1 Technologies utilisées

- ✓ ***Langage*** : C#
- ✓ ***Type d'application*** : Application Console
- ✓ ***Framework .NET :*** recommandé .NET 9 ou .NET 10
- ✓ ***IA Générative*** : pour le Le debugging .

3.2 Organisation du projet

Le projet est organisé autour de services indépendants, chacun responsable d'une partie de la logique métier :

- ➢ ***RessourceService***: gestion des ressources.
- ➢ ***ServiceReservation***: logique des réservations et gestion des conflits.

- ***ClientService :*** gestion des clients.
- ***ConfigurationService***: gestion des paramètres applicatifs.
- ***ConstraintService*** : validation et formatage des données.
- ***MenuPrincipal :*** composition des services et pilotage de l'interface utilisateur.

Cette organisation améliore la lisibilité du code, la maintenabilité et l'évolutivité du système.

4. Validation des données

La validation des données est centralisée dans ConstraintService :

- Utilisation d'expressions régulières pour vérifier les formats.
- Validation des emails selon des formats standards.
- Validation et formatage des numéros de téléphone.

Ce mécanisme garantit la cohérence et la fiabilité des données saisies par l'utilisateur.

5. Considérations spécifiques pour la France

Afin d'adapter l'application au contexte français, plusieurs points sont pris en compte :

5.1 Numéros de téléphone

- ✓ Adapter `GererContraintes/ConstraintService.cs pour accepter :
- ✓ Numéros français locaux à 10 chiffres : `0X XX XX XX XX`.
- ✓ Formats internationaux : `+33 X XX XX XX XX`.

5.2 Localisation

- Utilisation de la culture fr-FR pour :
- Les chaînes de caractères.
- Les formats de date et d'heure.
- Mise en œuvre via `CultureInfo`.

5.3 Fuseau horaire

Utilisation du fuseau horaire.

Emploi de types timezone-aware comme DateTimeOffset pour les réservations.

6.3 Intégration continue (CI/CD)

Mise en place d'un workflow GitHub Actions pour :

Build automatique.

Exécution des tests.

6.4 SDK ciblé

Préciser la version .NET 9 ou 10 via `global.json`.

7. Déploiement et installation (développeur)

- Système d'exploitation : Windows.
- IDE recommandée : JetBrains Rider.

- Exécution via CLI : `dotnet build` et `dotnet run`.

8. Métadonnées Git

- ✓ Branche principale : main
- ✓ Remote: origin
- ✓ Dépôt GitHub : MiniProjetsCSharp--CodeGenius

9. Conclusion

Ce projet de gestionnaire de réservations en application console répond efficacement aux besoins de gestion des ressources, des clients et des réservations à travers une architecture claire et orientée services. L'utilisation de services indépendants, d'un menu interactif et d'un mécanisme de validation centralisé garantit la fiabilité des données et la cohérence du fonctionnement global du système.

De plus, la prise en compte des aspects de qualité logicielle, de localisation et de conformité réglementaire (notamment le GDPR) renforce la crédibilité du projet et démontre une approche professionnelle du développement. Cette application constitue ainsi une base robuste et évolutive, pouvant être étendue vers une solution plus complète intégrant une interface graphique ou web, une persistance des données et des fonctionnalités avancées adaptées à un usage réel.