PROJET EF / ASP.NET / WEB API

INTRODUCTION

Vous êtes en charge de l'architecture logicielle d'une banque. La banque souhaite mettre en place des services web et des applications permettant d'afficher des informations sur les comptes bancaires, sur les cartes et sur les clients.

PARTIE 1 : ENTITY FRAMEWORK

Dans cette première étape vous devrez créer une architecture pour définir la base de données en utilisant Entity Framework Code First (c'est-à-dire la création de la base de données via le code).

SCHEMA BCR (BANK CUSTOMER RELATIONSHIP)

Pour commencer, vous devrez créer le schéma du modèle d'entités (diagramme de classes UML) à l'aide d'un logiciel ou version papier et le faire valider par le cher. Le schéma devra contenir les entités et respecter les directives métiers suivantes dictés par la banque :

- La banque gère 3 types de personnes : les clients, les employés et les managers.

- Informations à stocker pour :

    - Les clients

        - FirstName
        - LastName
        - DateOfBirth (DateTime)
        - Street (String)
        - ZipCode (String)
        - City

    - Les employés

        - FirstName
        - LastName
        - DateOfBirth (DateTime)
        - OfficeName (String)
        - IsJunior (bool)

    - Les managers

        - FirstName
        - LastName
        - DateOfBirth (DateTime)
        - OfficeName (String)
        - IsJunior (bool)

Toutes les colonnes décrites précédemment et crées par le schéma doivent être non-nullable en base de données

- Les managers dirigent plusieurs employés tandis que les employés peuvent avoir un manager.
- Les clients de la banque sont obligatoirement reliés à un conseiller qui est donc un employé de la banque.
- Les tables crées pour ces entités doivent avoir comme nom de schéma « BCR ».
- Vous devrez factoriser au mieux les entités de votre modèle.

SCHEMA CB (CORE BUSSINESS)

Vous devrez ensuite rajouter à ce schéma d'autres entités permettant de gérer les comptes bancaires et les cartes bancaires des clients.

- Il existe 2 types de comptes bancaires : les comptes de dépôt ou compte courant (Deposit) et comptes épargnes (Saving)
- Seuls les comptes de dépôt sont reliés à des cartes bancaires (Card), les comptes épargnes ne proposent pas cette possibilité.
- Un compte de dépôt peut avoir plusieurs cartes bancaires, mais une carte bancaire ne peut être reliée qu'à un seul compte de dépôt.
- Informations à stocker pour (*indique les informations requises) :

    - Les comptes de dépôt :

        - BankCode (string de 5 caractères)
        - BranchCode (string de 5 caractères)
        - AccountNumber (string de 11 caractères)
        - Key (string de 2 caractères)

Les 4 propriétés précédentes forment le BBAN ou aussi appelé RIB

- IBAN (string de 34 caractères)
- BIC (string de 11 caractères – *aussi appelé code SWIFT*)
- Balance *
- CreationDate (DateTime)
- AutorizedOverdraft *
- FreeOverdraft *
- OverdraftChargeRate (double – définissant le taux des agios)

*** - Les types utilisés par Balance (solde du compte), AutorizedOverdraft (découvert autorisé) et FreeOverdraft (découvert sans agios – sans frais supplémentaires) sont à définir.**

- Vous devrez garantir une certaine précision des données de l'application ainsi que de celles stockés en base.

- Les comptes épargnes :
    - BankCode (string de 5 caractères)
    - BranchCode (string de 5 caractères)
    - AccountNumber (string de 11 caractères)
    - Key (String de 2 caractères)

Les 4 propriétés précédentes forment le BBAN ou aussi appelé RIB

- IBAN (String de 34 caractères)
- BIC (String de 11 caractères – *aussi appelé code SWIFT*)
- Balance *

*** - Le type de données utilisé par Balance (solde du compte) est à définir.**
- Vous devrez garantir une certaine précision de l'information au sein de l'application ainsi qu'en base.

- MinimumAmount (int)
- MaximumAmount ** (int)
- InterestRate (double)
- MaximumDate ** (Datetime)

**** Les colonnes MaximumAmount et MaximumDate doivent être nullable en base de données, les comptes épargnes ne possèdant pas tous ces règles.**

- Les cartes bancaires :
    - NetworkIssuer (string – réseau du fournisseur de carte ex : Visa, Amex…)
    - CardNumber (string de 16 caractères)
    - SecurityCode (string de 4 caractères)
    - ExpirationDate (DateTime)

- Un client peut avoir plusieurs comptes bancaires, un compte bancaire ne peut avoir qu'un seul client propriétaire de compte.
- Les tables crées pour ces entités doivent avoir comme nom de schéma « CB ».
- Vous devrez factoriser au mieux les entités de votre modèle.

CODE

Après avoir défini votre schéma et l'avoir fait valider, vous devrez créer la ou les couches nécessaires à un accès à la base de données dans Visual Studio.

- Vous devrez définir un jeu de données minimum, les données doivent être insérées automatiquement dans la base via un « initializer ».

- Il sera possible de créer une application console pour tester votre modèle, mais ceci n'est pas forcément nécessaire.

- La chaine de connexion ainsi que la base de données doivent avoir comme nom « BankDb ».

PARTIE 2 : BACK OFFICE ASP.NET

Durant la première étape, vous avez créé une architecture pour définir la base de données en utilisant Entity Framework Code First. Après une revue de code approfondie de toute l'équipe, certaines parties ont pu légèrement changer. Vous avez donc normalement en votre possession une solution contenant un projet Dal (Data Access Layer) et un projet DomainModel (contenant les entités).

Dans cette deuxième étape, vous aurez à créer un site web et un services web qui permettront de dialoguer avec un site web et d'autres applications externes (que n'aurons donc pas à construire).

ASP.NET (BACK OFFICE)

Votre chef de projet vous demande de créer un site web qui sera utiliser uniquement en interne. Il permettra d'accéder et de modifier les informations de la base de données à travers la couche d'accès aux données. Il sera ainsi possible de modifier le contenu de chacune des entités (accéder/créer/éditer/supprimer).

Il faudra aussi gérer les droits d'accès à l'application. Les employés et les managers seront les seules à accéder à cette application.

- Un manager pourra accéder/créer/éditer/supprimer ses employées mais pas son supérieur.
- Un employée/manager pourra accéder/créer/éditer/supprimer un client, un compte ou même une carte

PARTIE 3 : FRONT OFFICE ASP.NET/ANGULAR

FRONT END INTERNE: WEB API + ANGULAR

Votre chef de projet vous demande de créer un site web qui sera utiliser uniquement en interne. Il permettra d'accéder et de modifier les informations de la base de données via une API. Il sera ainsi possible de modifier le contenu de chacune des entités (accéder/créer/éditer/supprimer).

Il faudra aussi gérer les droits d'accès à l'application. Les employés et les managers seront les seules à accéder à cette application.

· Un manager pourra accéder/créer/éditer/supprimer ses employées mais pas son supérieur.

· Un employée/manager pourra accéder/créer/éditer/supprimer un client, un compte ou même une carte

FRONT ENG CLIENT: ANGULAR

Dans un second temps, votre chef de projet vous demande de créer une interface web qui permettra aux clients d'avoir accès aux informations de leurs comptes. Pour cela, vous devez d'abord concevoir un projet Angular via l'@angular/cli afin de consommer la même API faite en .NET à l'étape précédente.

Pour simplifier le projet nous n'allons pas utiliser de système d'authentification, même si cela s'avère indispensable pour ce genre de service. Il serait ainsi possible de mettre en place une authentification par formulaire (FBA), par l'utilisateur (avec l'AD par exemple) ou par SSO.

Le service et le site devront permettre au client :

- De modifier ses informations personnelles
- De connaitre ses différents comptes au sein de la banque
- De savoir qui est son conseiller au sein de la banque
- De connaitre le solde de son compte
- De pouvoir imprimer un RIB soit BBAN + clé + IBAN + BIC
- De connaitre sa limite de découvert (« overdraft »), le taux d'intérêt de son découvert ainsi que son découvert sans agios
- D'accéder aux informations de solde minimum, maximum et taux d'intérêt des comptes d'épargne
- De faire un virement entre deux comptes de la banque qu'il s'agisse ou non du même client.

L'interface du site devra être très simple mais ergonomique, adapté pour pc, tablettes et smartphone.

PARTIE 4 : BI AVEC WEB API ET WPF

WPF

La dernière partie du projet consistera en la création d'une application WPF permettant d'afficher des informations clés sur les clients de la banque et de permettre au manager de prendre des décisions sur les offres que la banque proposera dans le futur.

Pour cela votre chef de projet souhaite la création d'un service Web Api qui permettra d'effectuer des calculs de statistiques sur un serveur et non sur une application cliente.

Les managers et la direction souhaitent avoir accès aux statistiques suivantes :

- Le nombre de clients de la banque
- Le nombre de clients par manager
- Les sommes épargnées par les clients de la banque
- Les sommes épargnées par les clients par manager
- Le solde total de l'ensemble des comptes de la banque
- Le pourcentage de clients qui possèdent une carte bancaire
- Le pourcentage de clients qui possèdent une carte bancaire par manager
- Le pourcentage de clients qui possèdent un compte d'épargne
- Le pourcentage de clients qui possèdent un compte d'épargne par manager

D'un point de vue technique :

- Le service Web Api :
    - Accèdera aux données à travers la couche DAL
    - Exécutera l'intégralité des calculs de statistiques

Pour tester votre Web Api, vous pourrez installer et utiliser le logiciel PostMan ou un logiciel similaire.

« BON CHANCE » ☺