Synthentic Financial Manager System For Fraud Transactions

v 1.1 Apr 2018

DESCRIPTION

Build a .Net web app that allow register bank transactions, mark a transaction as a fraud and search transactions according to one or more search criteria. You can use the dataset: https://www.kaggle.com/ntnu-testimon/paysim1 posted on Kaggle. It is described there.

PART I

Build a simple web application with a software architectural pattern Model–view–controller (MVC) to allow a user register transactions, mark a transaction as a fraud and search transactions according to the criteria listed below. Do not pay much attention to the UI/design side of things. The focus of the test should be back-end side of things. You can use Entity Framework or another ORM to allow the comunication between the web application and the database.

Be sure to allow these set of actions only to authenticated users and to users that match the following roles:

Role	Action
Assistant	Register bank transactions. Search transactions according to the criteria listed below.
Manager	Mark a transaction as a fraud.
Administrator	Full control.

Search criteria:

- IsFraud
- NameDest
- TransactionDate

Feel free to use the authentication method that you consider most appropriate.

(Framework version: 4.5 or 4.6 or .Net Core)

Design Rules

- The Controllers should **not** have direct dependency on the DbContext/Data access classes.
- Use Dependecy Injection/IoC container for dependency resolution

PART II

Create a simple Web API (REST service) that allows the consumer of such API to perform Assistant role operations (register and search). You need to include a security mechanism to the API to avoid unauthorized requests.

Feel free to serialize the Web Service responses in JSON or XML.

(Framework versión: 4.5 or 4.6 or .Net Core)

PART III

Propose and implement a **relational** database design in which will be stored the application data. You can populate your database with the Kaggle dataset.

Feel free to use the database engine with which you feel more familiar and stored procedures or functions if you consider most appropriate.

DELIVERABLES

- Complete source code and script of database in a public repository on **github** or **bitbucket**. If not possible please provide a .zip file but points will be deducted. *(required)*
- Readme file in the repository with all the steps to deploy and run the 3 components. (Web app, web service and database) and your contact information as a solution owner. *(required)*

* **Bonus points:**

- Clean and well documented code (each method and class)
- Unit tests for domain model
- Use of Cloud services (Azure/AWS/GCloud) to run the application
- Diagrams:
- Classes diagram of domain (business logic) model
- E-R diagram
- General system diagram (Components diagram)

zemoga®