Desafio .NET Concrete

Requisitos

- O projeto deverá abrir no Visual Studio 2015;
- Banco de dados LocalDB ou SqlServer Express;
- Gestão de dependências via gerenciador de pacotes NuGet;
- Utilização de Web API 2 como framework;
- Persistência com ADO.NET (Criação do Cadastro);
- Persistência com ferramenta de ORM (Entity Framework, nHibernate, etc...) para Login e Profile;
- Todos os endpoints devem trabalhar somente com JSON;
- Utilizar status codes de acordo com o tipo de resposta para a requisição.

Requisitos desejáveis

- JWT como token generator;
- Testes unitários;
- Cobertura de testes acima de 50%;
- Scripts desenvolvidos para uso no Postman para validar as APIs ou uma interface de usuário;
- Criptografia não reversível (hash) na senha e no token;
- Todas as respostas diferentes de sucesso devem retornar o seguinte conteúdo:

```
{
  "statusCode": 0, (StatusCode adequado)
  "mensagem": "sample string" (Mensagem adequada)
}
```

- A resposta de erro para endpoints não localizados também deverá retornar um 404;

Para refletir

Existem diversos Design Patterns, tais como Singleton, Template Method, DI, Observer, Memento, Repository, Facade, entre outros tantos. Favor desenvolver o desafio ponderando o equilíbrio entre os Design Patterns e a facilidade de entendimento.

Segue a documentação dos endpoints:

SignUp - Criação de Cadastro

- Este endpoint deverá receber um usuário com os seguintes campos:

```
{
  "nome": "sample string",
  "email": "sample string",
  "senha": "sample string",
  "telefones": [
    {
      "numero": "sample string",
      "ddd": "sample string"
    }
  ]
}
```

Em caso de sucesso retornar os dados do usuário cadastrado e incluir os seguintes campos:

- "id": id do usuário (pode ser o próprio gerado pelo banco, porém seria interessante se fosse um GUID);
- "data_criacao": data da criação do usuário;
- "data_atualizacao": data da última atualização do usuário;
- "ultimo_login": data do último login (no caso da criação, será a mesma que a criação);
- "token": token de acesso da API (pode ser um GUID ou um JWT);
- Caso o e-mail já exista, deverá retornar o status code http correto com a mensagem "E-mail já existente";
- O token deverá ser persistido junto com o usuário.

Login

- Este endpoint irá receber um objeto com os seguintes campos:

```
{
  "email": "sample string",
  "senha": "sample string"
}
```

- Caso o e-mail e a senha estejam corretas, retornar os mesmos dados do endpoint de SignUp;
- Caso o e-mail não exista, retornar o status code http apropriado mais a mensagem "Usuário e/ou senha inválidos";

- Caso o e-mail exista mas a senha não bata, retornar o status code http apropriado mais a mensagem "Usuário e/ou senha inválidos";
- Atualizar o campo de ultimo_login deste usuário para a data atual;

Profile

- Chamadas para este endpoint devem conter um header na requisição de Authentication com o valor "Bearer {token}" onde {token} é o valor do recebido através do SignUp ou Login de um usuário;
- Caso o token não exista, retornar o status code http apropriado com a mensagem "Não autorizado";
- Caso o token exista, buscar o usuário pelo id passado e comparar se o token do usuário encontrado é igual ao token passado no header:
    - Caso não seja o mesmo token, retornar erro com status code http apropriado e mensagem "Não autorizado";
- Caso seja o mesmo token, verificar se o último login foi a MENOS que 30 minutos atrás:
    - Caso não seja a MENOS que 30 minutos atrás, retornar erro com status code http apropriado com mensagem "Sessão inválida";
- Caso tudo esteja ok, retornar os dados do usuário;

Entrega do Desafio

O projeto criado para o desafio deve ser compactado e enviado de volta para a Concrete através da pessoa que deu início ao contato com você.