Desafio MB

Introdução

Aplicativo de controle de eventos Acadêmicos e Corporativos via Web, com comunicação com aplicativos mobile.

Caracterização dos Problemas

- Login por meio de Usuário e Senha, Linkedin, Facebook.
    - Cadastro pelos serviços secundários permite obter alguns dados como telefone, imagem de perfil, endereço, e e-mail dos usuários.
- Criptografia de dados confidenciais(Senha, Dados Bancários) por AES 256.
    - É necessário uma chave para *encrypt* e *decrypt*.
- Cadastrar/Organizar Eventos.
- Efetuar Pagamento de Ingressos via cartão de crédito.
- Validar entrada de usuário em evento.
- Recomendar Eventos a usuários.
- Áreas separadas para cadastro de Evento e Administração.
- Cadastrar Usuários:
    - Usuários do tipo ‘Gerenciador de Eventos’, ‘Administrador’, ‘Visitante’.
    - Cada um com níveis de permissões específicas como (Criar, Alterar, Desativar, etc.).

Metodologia e Ferramentas

- Google Docs
- Draw.io
- Visual Studio ASP.NET/C#
- Fiddler 4

Funcionalidades

Observação: Quando dizemos ‘desativada’ entenda-se excluída, sendo que como é necessário manter a consistência do banco de dados, nunca realmente deletamos informações do banco de dados, apenas a tornamos desabilitada.

Os vários tipos de usuários são:

- Visitante:
    - Cadastro próprio.
    - Cria ingressos para eventos.
    - Visualizar Eventos.
- Administrador Nível 1:
    - Cria, Altera, e Desativa Gerente de Eventos Nível 1.
    - Cria, Altera, e Desativa Gerente de Eventos Nível 2.
- Administrador Nível 2:
    - Cria, Altera, e Desativa Gerente de Eventos Nível 1.
    - Cria, Altera, e Desativa Gerente de Eventos Nível 2.
    - Cria, Altera, e Desativa Administradores Nível 1.
    - Cria, Altera, e Desativa Administradores Nível 2.
    - Cria, Altera, e Desativa Visitantes.
    - Cria, Altera, e Desativa Eventos.
    - Defini se um ingresso foi utilizado.
- Gerente de Eventos Nível 1:
    - Defini se um ingresso foi utilizado.
    - Altera, Eventos.
- Gerente de Eventos Nível 2:
    - Defini se um ingresso foi utilizado.
    - Cria, Altera, e Desativa Eventos.
    - Cria, Altera, e Desativa Gerente de Eventos Nível 1.

Processo 1: Cadastro de Usuário

Caso seja Visitante, o próprio irá efetuar:

- Efetuar Login por Facebook ou Linkedin, se for o caso.
- Caso seja Login com link a conta externa, os seguintes dados podem ser obtidos do *Profíle* deste link: Endereço, Telefone, Email, Foto de *Profíle*.
- Inserção de dados cadastrais: RG, CPF, Data de Nascimento, etc., inclusive dados que não forem obtidos de uma conta externa se for este o caso.
- Inserção de dados de pagamento: Catão de Crédito, Endereço de cobrança, tipo de cartão, e etc.

- É Necessário uma validação de dados do cartão (cada tipo de cartão possui uma função de validação diferente.[1])
- Também é necessário validar o CPF.[2]
- CVV de Cartão de Crédito não pode ser guardado em Banco de dados por motivos de segurança, e será necessário ser inserido manualmente a cada compra de ingresso.
- Criptografia de dados pessoais.

Caso seja Administrador Nível 2:

- Primeiro Administrador Nível 2 obrigatoriamente deve existir na aplicação desde a primeira execução (pré-definido em BD).
- Irá obter uma lista de outros usuários para modificar.
- Pode criar novos usuários dos tipos:
    - Visitante
    - Administrador Nível 1
    - Administrador Nível 2
    - Gerente de Eventos Nível 2
    - Gerente de Eventos Nível 2

Caso seja Administrador Nível 1:

- Irá obter uma lista de outros usuários do tipo Gerente de Eventos Nível 1 e 2 para modificar.
- Pode criar novos usuários dos tipos:
    - Gerente de Eventos Nível 1
    - Gerente de Eventos Nível 2

Caso seja Gerente de Eventos Nível 2:

- Irá receber uma lista de Gerente de Eventos Nível 1 para modificar.
- Pode criar novos usuários do tipo Gerente de Eventos Nível 1.

Processo 2: Cadastro de Evento

Caso seja Visitante:

- Poderá visualizar todos os eventos disponíveis.
- Buscar Evento por nome ou descrição por meio de consulta.
- Exibir dados do pagamento do Ingresso, caso exista.

Caso seja Gerente de Eventos Nível 2:

- Irá receber uma lista de eventos criados por este.
- Poderá alterar ou desativar um evento.
- Ao criar um evento, as informações relevantes ao evento devem ser inseridas: Nome do evento, Endereço, Data do evento, Quantidade máxima de visitantes se aplicável, e uma descrição sobre o evento.

---

[1] veja: https://www.freeformatter.com/credit-card-number-generator-validator.html
[2] veja: https://dicasdeprogramacao.com.br/algoritmo-para-validar-cpf/

Caso seja Gerente de Eventos Nível 1:

- Irá receber uma lista de eventos criados por este.
- Poderá alterar dados de um evento.

Caso seja Administrador Nível 2:

- Irá receber uma lista de eventos criados por todos os Gerentes de Eventos Nível 2.
- Poderá alterar ou desativar um evento.
- Ao criar um evento, as informações relevantes ao evento devem ser inseridas: Nome do evento, Endereço, Data do evento, Quantidade máxima de visitantes se aplicável, e uma descrição sobre o evento.
    - Inclusive a qual gerente pertence o Evento.

Processo 3: Validar Participação em Evento

Caso seja Gerente de Eventos Nível 2, ou 1, ou Administrador Nível 2:

- Dado um ingresso o Gerente de Eventos pode:
    - Confirmar se o ingresso realmente pertence ao evento em questão.
    - Definir que o ingresso foi utilizado para a entrada do evento.

Processo 4: Login de Usuário

Caso seja Visitante:

- Caso seja Login com link de Linkedin ou Facebook:
    - Verificamos se a aplicação externa permite login pelo nosso aplicativo.
    - Verificamos se existe um token válido ou se é necessário a renovação deste.
    - Finalmente criamos uma nova sessão para este usuário automaticamente.
    - Caso contrário informamos o tipo de erro, como por exemplo: Permissão deste aplicativo removida, ou falha na comunicação com serviço externo, etc.
- Caso seja Usuário e Senha:
    - Comparamos com o Usuário e Senha em Banco.
        - A senha provida seria criptografada e o resultado comparado com o em banco, para garantir que nenhum dado confidencial seja revelado.
    - Se correto criamos a sessão do Visitante, ou informamos um erro de login.

Caso seja Gerente de Eventos Nível 2 , ou 1, ou Administrador Nível 2 ou 1:

- Login com Usuário e Senha somente.
- Acesso às áreas baseado no tipo de usuário.

Processo 5: Comprar Ingresso

Caso seja Visitante:

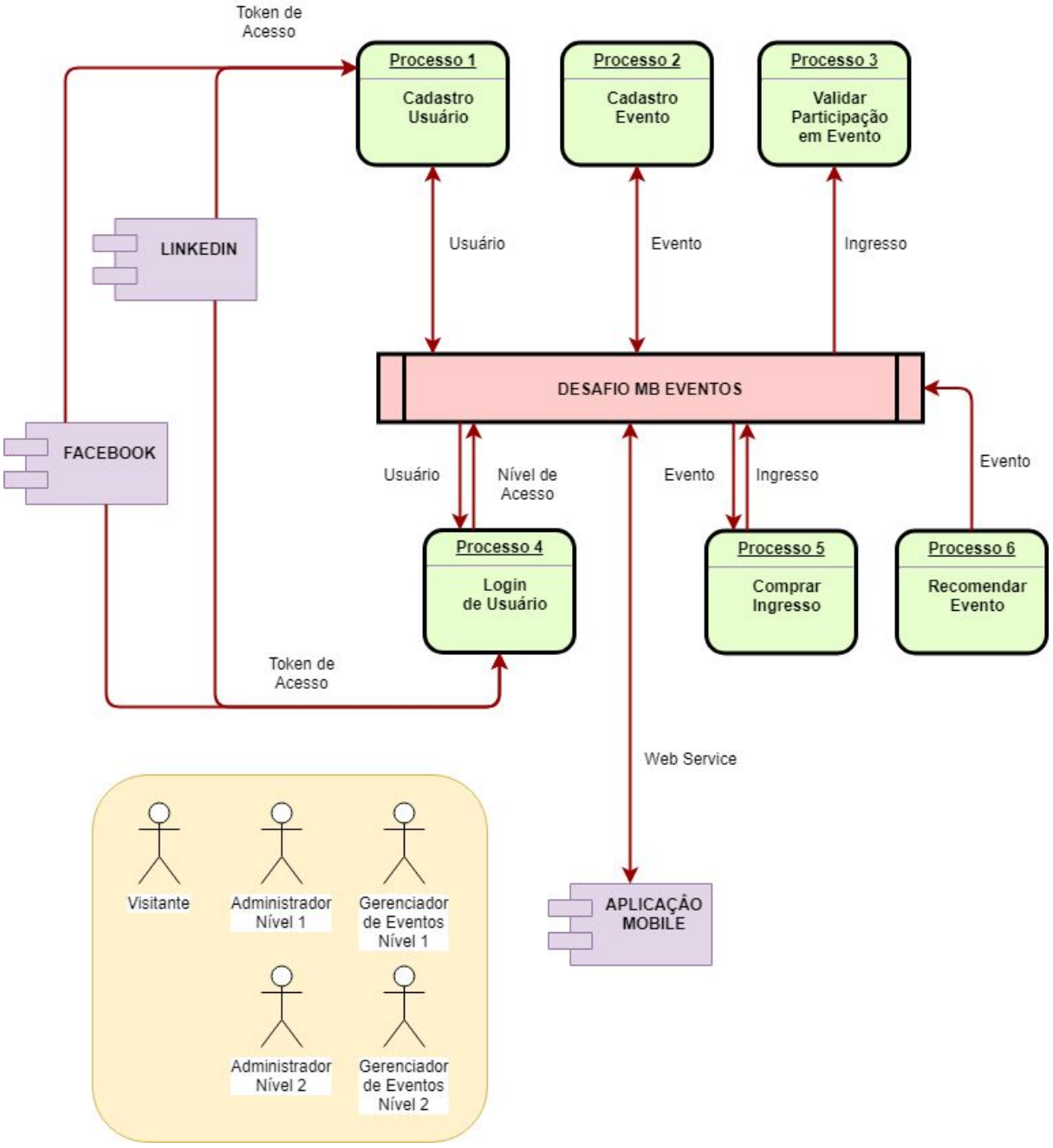
- O Visitante irá ver os dados do Evento novamente para confirmar.
- Dado o Evento, o Visitante deverá informar o Código de Validação do Cartão de Crédito (CVV) para efetuar a compra.
- A compra gera um Ingresso, que será utilizado na entrada do Evento.
    - Para saber se já existe um Ingresso é necessário navegar até a página do Evento no sistema.

Processo 6: Recomendar Eventos

Caso seja Visitante:

- Ao entrar na aplicação, caso já exista um ou mais Ingressos relacionado ao Visitante, Obtemos as palavras chave das descrições de cada evento relacionados aos Ingressos.
- Retornamos uma busca de Eventos com as três palavras chave encontradas com maior frequência, logo no início da aplicação.
- Podemos sugerir os demais eventos do mesmo Gerenciador de Eventos do último Ingresso para o Visitante.

Workflow


Token de Acesso
Processo 1
Cadastro Usuário
Processo 2
Cadastro Evento
Processo 3
Validar Participação em Evento
LINKEDIN
Usuário
Evento
Ingresso
DESAFIO MB EVENTOS
FACEBOOK
Usuário
Nível de Acesso
Evento
Ingresso
Evento
Processo 4
Login de Usuário
Processo 5
Comprar Ingresso
Processo 6
Recomendar Evento
Token de Acesso
Web Service
Visitante
Administrador Nível 1
Gerenciador de Eventos Nível 1
Administrador Nível 2
Gerenciador de Eventos Nível 2
APLICAÇÃO MOBILE

Tabelas

Observações:
Informações em (info) são meramente comentários sobre o campo.
Informações em [info] são condições sobre o campo.

Tabela Usuários

id [key]
RG [key?]
CPF [key?]
Endereço_Rua
Endereço_Bairro
Endereço_Complemento
Endereço_CEP
Endereço_Cidade
Endereço_Estado
Endereço_País
E-mail
Data_Nascimento (para obter idade)
Sexo
conta_ativa (nunca deletar uma conta!) [ativa, desativada]
Nome
Tipo_login [usr/senha, linkedin, facebook]
Imagem_profile_path (se houver, do linkedin e facebook)
Last_token [facebook, linkedin]
Login
Senha [criptografia AES 256]
Tipo_acesso [Administrador, Visitante, Gerenciador de Evento]
Numero_Cartão
Nome_Titular
Data_Expiração
Tipo_Cartão [visa, mastercard, etc]
Endereço_de_Cobrança_Rua
Endereço_de_Cobrança_Bairro
Endereço_de_Cobrança_Complemento
Endereço_de_Cobrança_CEP
Endereço_de_Cobrança_Cidade
Endereço_de_Cobrança_Estado
Endereço_de_Cobrança_País
Session_Id
Subordinacao (indica quem é o Gerente superior)

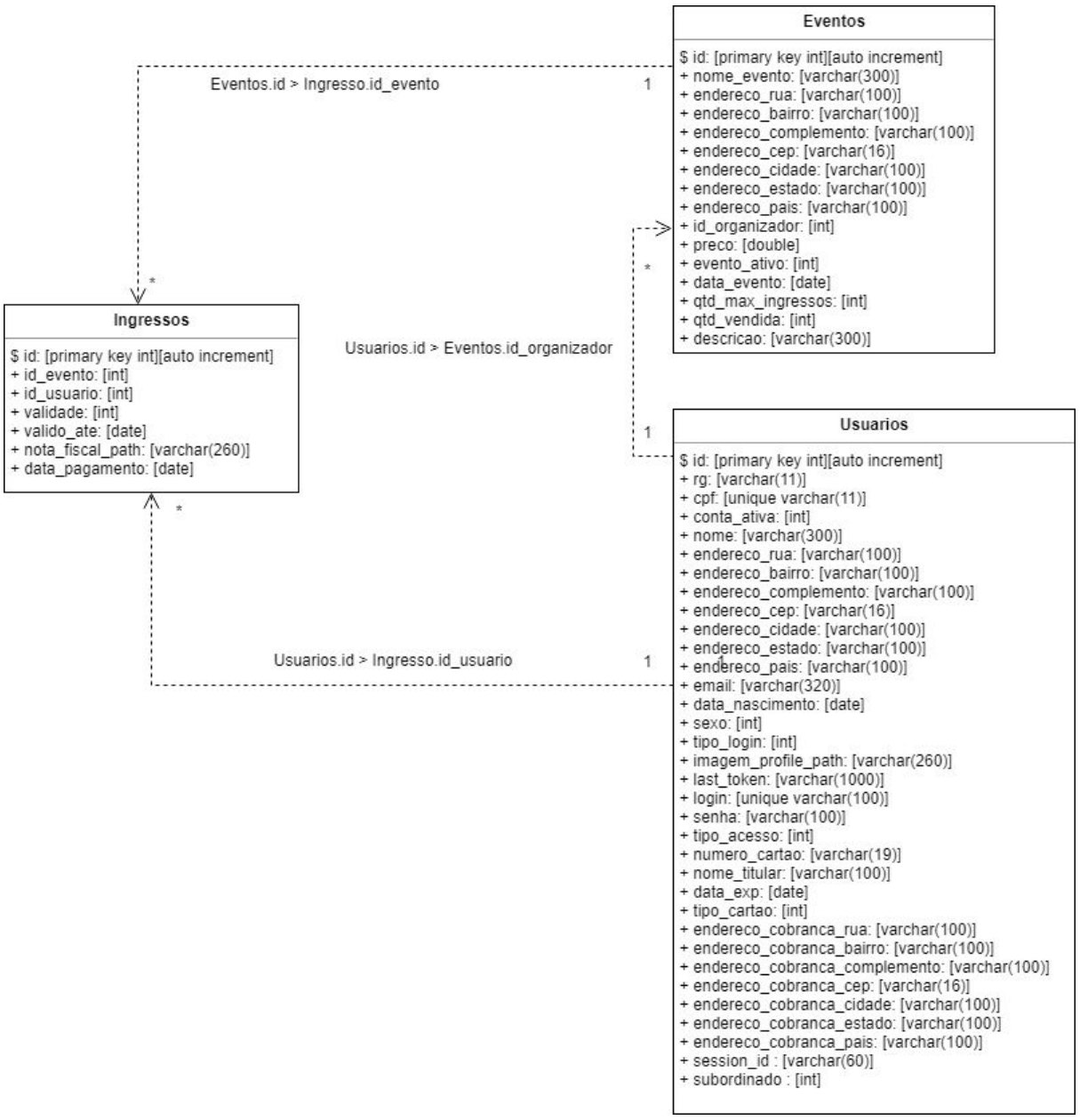
Tabela Eventos

id [key]
Nome
Endereço_Rua
Endereço_Bairro
Endereço_Complemento
Endereço_CEP
Endereço_Cidade
Endereço_Estado
Endereço_País
Organizador (usuário)
Preço
Evento_ativo [ativo, desativado]
Data_Evento
Quantidade_máxima_de_ingressos (evitar lotação?)
Quantidade_vendida
Descrição (Informações sobre o evento)

Tabela Ingresso

id [key]
id_evento [import key]
id_usuário [import key]
validade [Esperando_pagamento, Pronto_para_usar, Usado]
valido_até [data](data de expiração do ingresso)
nota_fiscal_path (arquivo xml)
data_pagamento [data]

UML de Banco de Dados



Funcionalidades Implementadas

A implementação mantém conexões com o banco de dados desenvolvido em PostgreSQL via NHibernate, e funciona via comunicações POST com clientes capazes de enviar HTTP POSTs contendo dados no formato JSON.

Os testes foram realizados utilizando a ferramenta Fiddler 4, que permite a criação destes POSTs e mantém um histórico destes, para facilitar testes de API Web.

A aplicação adere aos requisitos desejados de funcionalidade previstos na caracterização de problemas, exceto pelas funcionalidades de criptografia de dados e login via Facebook e Linkedin, visto que o desejo era de um aplicativo para prova de conceito.

URIs e JSON Requests

Módulo de Usuários (UsuarioController)

Listar Usuários

POST uri: desafiomb/usuario/listar
json:

```
{
   "sessionID":"hrgKdeHv1t"
}
```

result:

```
{
   "list":
     [ {"id":1,
     ...
     "subodinado":0 } ]
}
```

Criação/Alteração de Visitante

obs: para fazer alguma alteração em um visitante, basta inserir a sessionid do visitante no respectivo campo e as alterações serão feitas baseado na id de sessão.

POST uri: desafiomb/usuario/visitante
json:

```
{
   "sessionid":null,
   "usr":{
      "id":0,
      "rg":"382572749",
      ...
      "subordinado":0,
   }
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Criação de Usuário

POST uri: desafiomb/usuario/criacao
json:

```
{
   "sessionID":"hrgKdeHv1t",
   "usr":{
      "id":0,
      "rg":"382572749",
      ...
      "subordinado":0,
   }
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Alteração de Usuário

POST uri: desafiomb/usuario/atualizar
json:

```
{
   "sessionID":"hrgKdeHv1t",
   "usr":{
      "id":0,
      "rg":"382572749",
      ...
      "subordinado":0,
   }
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Desativação de Usuário

POST uri: desafiomb/usuario/desabilitar
json:

```
{
   "sessionid":"8ZMfsxFNCE",
   "usrId":4
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Módulo de Eventos (EventoController)

Listar Eventos

obs: listingresso é a lista com o id de cada ingresso para cada item da lista de eventos, caso não exista o valor será -1. A lista terá os Eventos que o usuário pode ver, dependendo do nível de acesso da sessão (Visitante, Gerente de Eventos, Administrador, etc).

POST uri: desafiomb/evento/visitante
json:

```
{
   "sessionID":"hrgKdeHv1t"
}
```

result:

```
{
   "list":
     [ {"id":1,
        "nomeEvento":"Evento",
        ...
        "descricao":"Evento sobre..." } ],
   "listingresso":
     [ 5,
     ...
     -1 ]
}
```

Criação de Eventos

POST uri: desafiomb/evento/criacao
json:

```
{
   "sessionid":"hrgKdeHv1t",
   "evt":{
      "id":0,
      "nomeEvento":"Evento",
      ...
      "descricao":"Evento sobre...",
   }
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Alteração de Evento

POST uri: desafiomb/evento/alterar
json:

```
{
   "sessionid":"hrgKdeHv1t",
   "evt":{
      "id":5,
      "nomeEvento":"Evento",
      ...
      "descricao":"Evento sobre...",
   }
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Desativação de Evento

POST uri: desafiomb/evento/desabilitar
json:

```
{
   "sessionid":"8ZMfsxFNCE",
   "evtId":4
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Módulo de Validação de Ingressos (ValidarIngressoController)

Gerente valida o ingresso de um Visitante

POST uri: desafiomb/validaringresso
json:

```
{
   "sessionid":"NNpEiW0wgG",
   "ingressoid":2
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Módulo de Login (LoginController)

Login

POST uri:
json:

```
{
   "usuario":"admin",
   "senha":"123456"
}
```

result:

```
{
   "result":"Login OK.",
   "sessionID":"hrgKdeHv1t"
}
```

Logout

POST uri: desafiomb/logout
json:

```
{
   "sessionID":"hrgKdeHv1t"
}
```

result:

```
{
   "result":"Logout OK.",
   "sessionID":null
}
```

Módulo de Compra de Ingresso (IngressoController)

Compra de Ingresso para um Evento

POST uri: desafiomb/ingresso/comprar
json:

```
{
   "sessionid":"ni45ZVMZCR",
   "eventid":3,
   "cvv":"462"
}
```

result:

```
{
   "result":"<Resultado da operação>"
}
```

Módulo de Recomendação (RecomendacaoController)

Obter recomendações

POST uri: desafiomb/evento/recomendacao
json:

```
{
   "sessionID":"hrgKdeHv1t"
}
```

result:

```
{
   "result":
     [ {"id":1,
        "nomeEvento":"Evento",
        ...
        "descricao":"Evento sobre..." } ]
}
```

Conclusão

Foram estudados os seguintes conceitos durante a implementação desta aplicação:

- Requests via POST/GET
- Aplicações .NET Core
- NuGet NHibernate
- NuGet Pgsql
- Mapeamento NHibernate

Conclusão