|  |
| --- |
| Instituto Infnet – Faculdade de Tecnologia |
| PROJETO DE BLOCO |
| Desenvolvimento .NET |

|  |
| --- |
| Vinicius Mansoldo Walsh Ferreira  30/11/2020  Matrícula 14911955767  Análise e Desenvolvimento de Sistemas |
|  |

Sumário

[Introdução 2](#_Toc486642269)

[Objetivo 2](#_Toc486642270)

[A rede social 2](#_Toc486642271)

[Realizado 2](#_Toc486642272)

[Modelagem estrutural e dinâmica em alto nível de abstração 3](#_Toc486642273)

[Modelagem com diagramas de classe 3](#_Toc486642274)

[Casos de uso 3](#_Toc486642275)

[Caso de uso 01 – Registro de uma nova conta 3](#_Toc486642276)

[Caso de uso 02 – Login 4](#_Toc486642277)

[Caso de uso 03 – Edição do perfil 4](#_Toc486642278)

[Caso de uso 04 – Postagem pela página de detalhes 5](#_Toc486642279)

[Caso de uso 05 – Postagem pela página inicial 5](#_Toc486642280)

[Caso de uso 06 – Acessar o perfil de outro usuário através da página inicial 6](#_Toc486642281)

[Caso de uso 07 – Desconectar da rede social 6](#_Toc486642282)

[Visão geral de interação 7](#_Toc486642283)

[Registro de uma nova conta 7](#_Toc486642284)

[Login 7](#_Toc486642285)

[Fluxo de comunicação entre camadas 8](#_Toc486642286)

[Requisitos de software 8](#_Toc486642287)

[Escopo do projeto 8](#_Toc486642288)

[Análise de requisitos 8](#_Toc486642289)

[Refinamento estrutural 10](#_Toc486642290)

[Identificação dos atributos das classes 10](#_Toc486642291)

[ProfileModel – A classe do perfil social 10](#_Toc486642292)

[PostModel – A classe das postagens 11](#_Toc486642293)

[AppContext – Code first 12](#_Toc486642294)

[Banco de dados 12](#_Toc486642295)

[Tabelas 12](#_Toc486642296)

[Conta 12](#_Toc486642297)

[Perfil 13](#_Toc486642298)

[Post 13](#_Toc486642299)

# Introdução

## Objetivo

Este documento tem como fim apresentar a estrutura do que foi desenvolvido para o Projeto de Bloco (*Assessment*), no qual o objetivo era o desenvolvimento de uma rede social.

## A rede social

Com o propósito de reunir usuários sob o contexto de divulgarem conteúdo sobre suas respectivas hortas, era necessário que a rede social possuísse os seguintes itens:

* Página de perfil;
  + Edição do próprio perfil;
  + Visualização de perfis (próprio e de terceiros);
* Página para publicação de conteúdo (textual e/ou imagens);
* Relacionamento entre usuários;
  + Visualização de conteúdo gerado por terceiros;
  + Relacionamento de amizade e/ou seguidores (opcional);
* Telas de controle de acesso;
  + Registro;
  + Login.

Através desses itens, o usuário poderia, mediante cadastro, publicar conteúdo e visualizar conteúdo publicado por outros usuários.

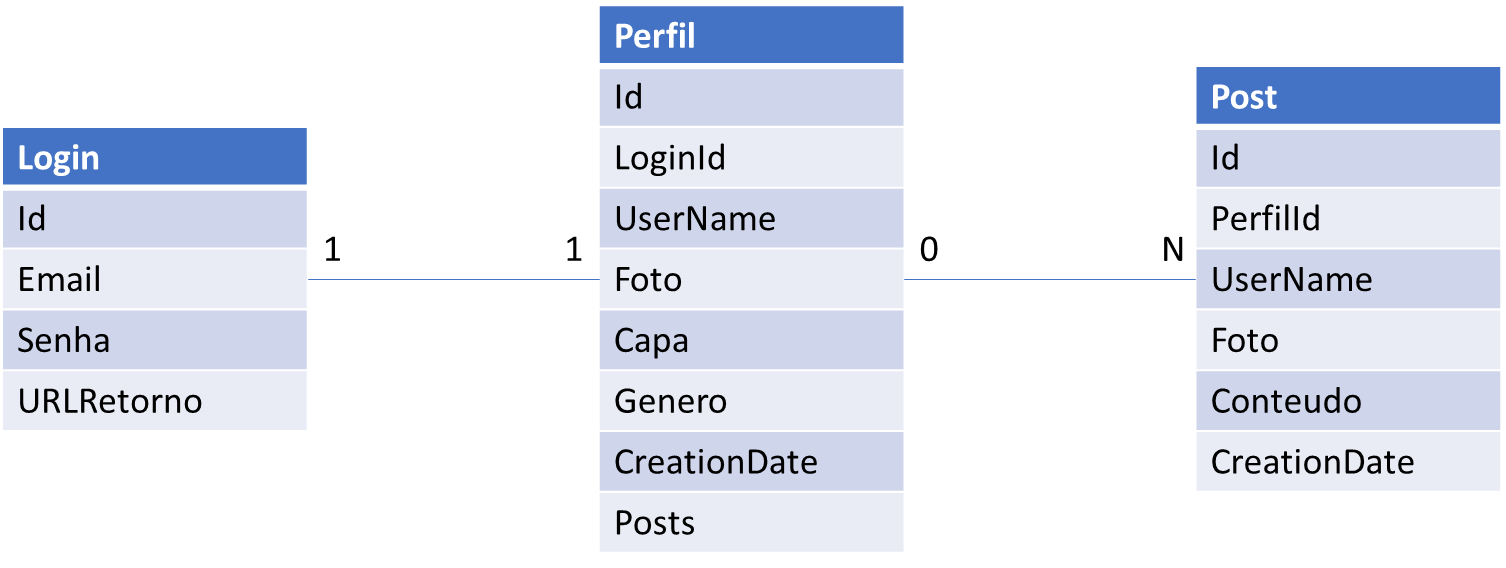
## Realizado

O projeto entregue contou com as seguintes *features* em estado funcionais:

* Página de perfil;
  + Edição do próprio perfil;
    - Edição do nome de usuário, foto de perfil, foto de capa e gênero;
    - Informações como e-mail de cadastro e data de criação do perfil não são editáveis;
  + Visualização de perfis (próprio e de terceiros);
    - Visualização de postagens do dono do perfil sendo visualizado;
    - Caso o perfil visualizado seja o próprio do usuário logado, o mesmo pode realizar a publicação de novas postagens;
* *Feed*, no qual é possível visualizar postagens (suas e de terceiros)
  + As postagens podem conter texto, imagem ou ambos;
  + Cada postagem mostra quem foi seu criador, gerando um link para sua página de perfil;
  + Os posts são exibidos cronologicamente, do mais novo até o mais antigo;
* Relacionamento entre usuários;
  + Visualização de conteúdo gerado por terceiros;
  + Visualização de outros perfis;
* Telas de controle de acesso;
  + Registro;
    - Proteção contra e-mails inválidos (e.g.: sem “@” ou “.”);
    - Durante o registro, é enviado um e-mail de confirmação de criação de conta para o e-mail fornecido no ato de cadastro;
  + Login;
    - A não ser que o usuário esteja conectado, não é possível visualizar ou gerar qualquer tipo de conteúdo na rede social.

# Modelagem estrutural e dinâmica em alto nível de abstração

## Modelagem com diagramas de classe



Figure

## Casos de uso

### Caso de uso 01 – Registro de uma nova conta

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Nenhuma

#### Fluxo básico:

1. O usuário fornece e-mail, senha e confirmações
2. Os dados do usuário são validados, encriptados e persistidos pelo sistema
3. O sistema gera e envia um link, para que o usuário confirme seu e-mail
4. O usuário confirma seu e-mail, clicando no link
5. O sistema confirma que a conta do usuário foi criada com sucesso
6. O sistema redireciona o usuário para a página de edição do próprio perfil

#### Fluxos alternativos:

1. O usuário deixa de fornecer algum campo de registro
   1. É exibido uma mensagem por campo solicitando que sejam preenchidos todos os campos
2. O e-mail ou a senha do usuário não são válidos
   1. O sistema exibe uma mensagem de erro por campo para o usuário

#### Regras de validação:

* Validez do e-mail
  + Deve ser composto por: caracteres -> @ -> caracteres -> . -> caracteres
    - Informações complementares após o primeiro ponto são possíveis (e.g.: “.br” de “.com.br”)
* Força da senha
  + Deve conter pelo menos um caractere maiúsculo, minúsculo, especial e numeral
* Confirmação de senha
  + A confirmação de senha deve ser igual à senha

### Caso de uso 02 – Login

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Conta previamente registrada

#### Fluxo básico:

1. O usuário fornece e-mail e senha
2. Os dados do usuário são validados de acordo com o que está registrado no banco de dados
3. Após conferir os dados, é permitido o login do usuário, realizando o redirecionamento para a página principal

#### Fluxos alternativos:

1. O usuário deixa de fornecer algum campo de registro
   1. É exibido uma mensagem por campo solicitando que sejam preenchidos todos os campos
2. O e-mail ou a senha do usuário não são válidos
   1. O sistema exibe uma mensagem de erro por campo para o usuário

#### Regras de validação:

* Validez do e-mail
  + Deve ser composto por: caracteres -> @ -> caracteres -> . -> caracteres
    - Informações complementares após o primeiro ponto são possíveis (e.g.: “.br” de “.com.br”)
* Os registro deve existir no banco de dados

### Caso de uso 03 – Edição do perfil

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Possuir uma conta registrada na rede social

#### Fluxo básico:

1. O usuário fornece seu nome de usuário, fotos de perfil e capa e informa seu gênero
2. O usuário clica em “Salvar” para persistir as informações
3. O sistema substitui os dados anteriores pelos dados informados
4. O sistema redireciona o usuário para sua página de perfil

#### Fluxos alternativos:

1. O usuário deixa a página sem clicar em “Salvar”
   1. O sistema não considera as informações fornecidas anteriormente
2. O usuário tenta passar parâmetros pela URL para editar outro perfil
   1. O sistema redireciona o usuário para a tela de edição do seu próprio perfil

#### Regras de validação:

* Nome de usuário não pode ser deixado vazio
  + O sistema exibe uma mensagem informando que o campo não pode ser deixado em branco
* O ID passado pela URL deve ser correspondente ao ID do usuário conectado

### Caso de uso 04 – Postagem pela página de detalhes

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Possuir uma conta registrada na rede social
* Estar na sua própria página de detalhes

#### Fluxo básico:

1. O usuário insere uma mensagem de texto no campo “Diga o que está pensando...”
2. O usuário insere uma imagem
3. O usuário clica em “Criar”
4. O sistema redireciona o usuário para a página inicial, a qual contém todos os posts

#### Fluxos alternativos:

1. O usuário não insere uma mensagem
   1. O sistema irá postar apenas a imagem inserida
2. O usuário não insere uma imagem
   1. O sistema irá postar apenas o texto inserido
3. O usuário não insere uma mensagem e nem uma imagem
   1. O sistema irá informar que não é possível inserir uma postagem sem dados

#### Regras de validação:

* Todos os campos não podem estar vazios simultaneamente

### Caso de uso 05 – Postagem pela página inicial

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Possuir uma conta registrada na rede social

#### Fluxo básico:

1. O usuário insere uma mensagem de texto no campo “Diga o que está pensando...”
2. O usuário insere uma imagem
3. O usuário clica em “Criar”
4. O sistema atualiza a página, exibindo novamente todas as postagens (incluindo a que foi criada no momento)

#### Fluxos alternativos:

1. O usuário não insere uma mensagem
   1. O sistema irá postar apenas a imagem inserida
2. O usuário não insere uma imagem
   1. O sistema irá postar apenas o texto inserido
3. O usuário não insere uma mensagem e nem uma imagem
   1. O sistema irá informar que não é possível inserir uma postagem sem dados

#### Regras de validação:

* Todos os campos não podem estar vazios simultaneamente

### Caso de uso 06 – Acessar o perfil de outro usuário através da página inicial

#### Atores:

* Usuário 1
* Outro usuário (usuário 2)
* Sistema

#### Pré-condições:

* Possuir uma conta registrada na rede social
* Usuário 2 deve ter realizado alguma postagem

#### Fluxo básico:

1. O usuário clica no botão “Perfil” do usuário 2
2. O sistema redireciona o usuário 1 para a tela de detalhes do usuário 2

#### Fluxos alternativos:

1. O usuário 1 clica na foto de perfil do usuário 2
   1. O sistema redireciona o usuário 1 para a tela de detalhes do usuário 2
2. O usuário 1 clica no nome do usuário 2
   1. O sistema redireciona o usuário 1 para a tela de detalhes do usuário 2

### Caso de uso 07 – Desconectar da rede social

#### Atores:

* Usuário
* Sistema

#### Pré-condições:

* Possuir uma conta registrada na rede social
* Estar conectado

#### Fluxo básico:

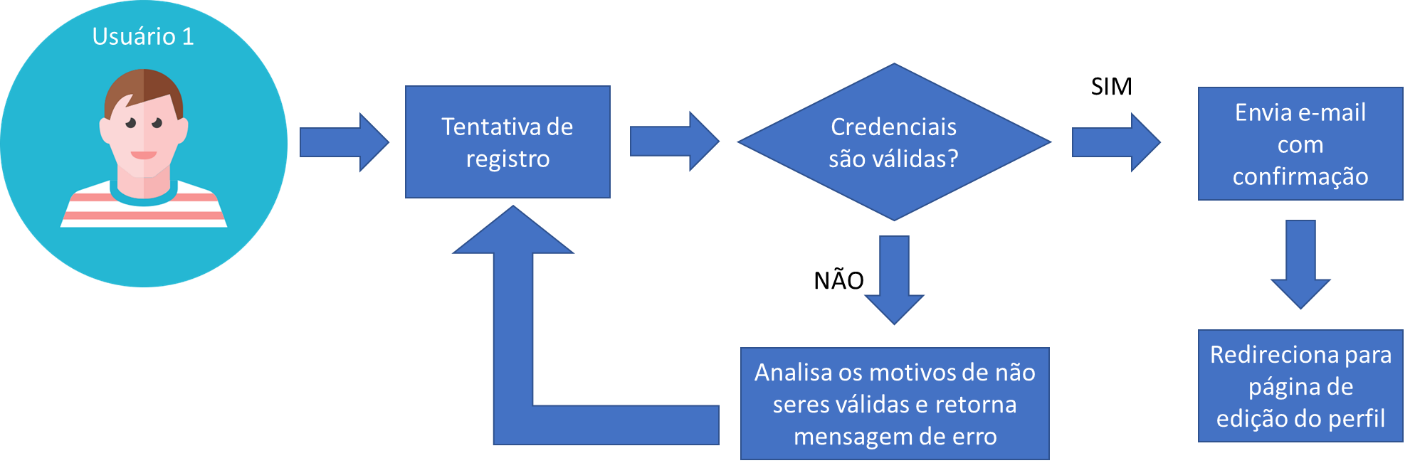
1. Em qualquer página do sistema, o usuário clica em “Sair” na barra de navegação

#### Fluxos alternativos:

1. O usuário fecha a janela do *browser*
   1. O usuário não será desconectado

## Visão geral de interação

### Registro de uma nova conta

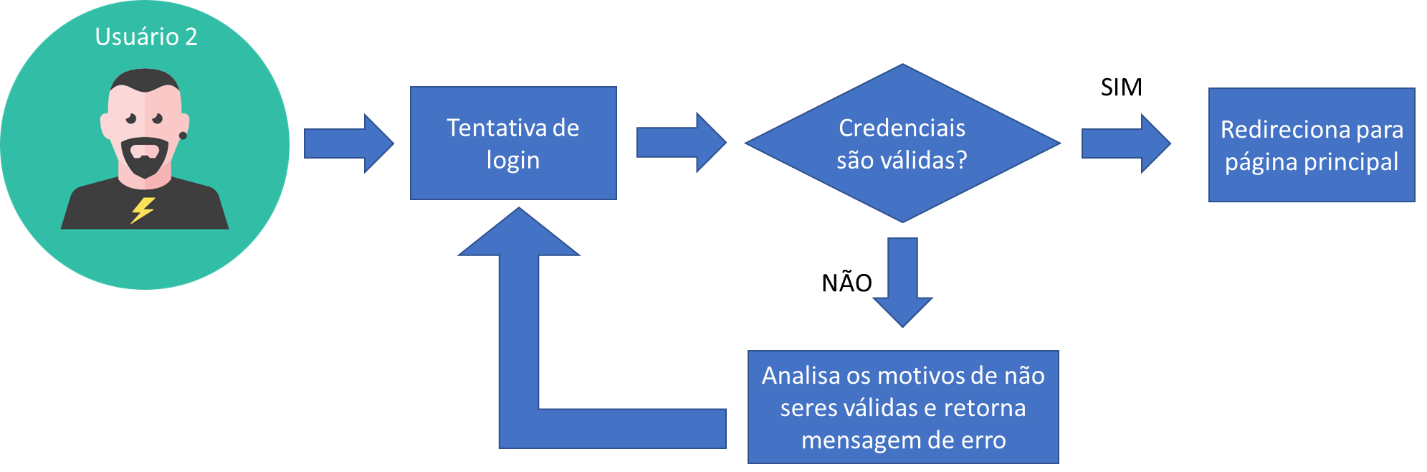


Figure

Através do fluxo apresentado na *figure 2*, pode-se observar o que foi abordado no **Caso de uso 01 – Registro de uma nova conta**.

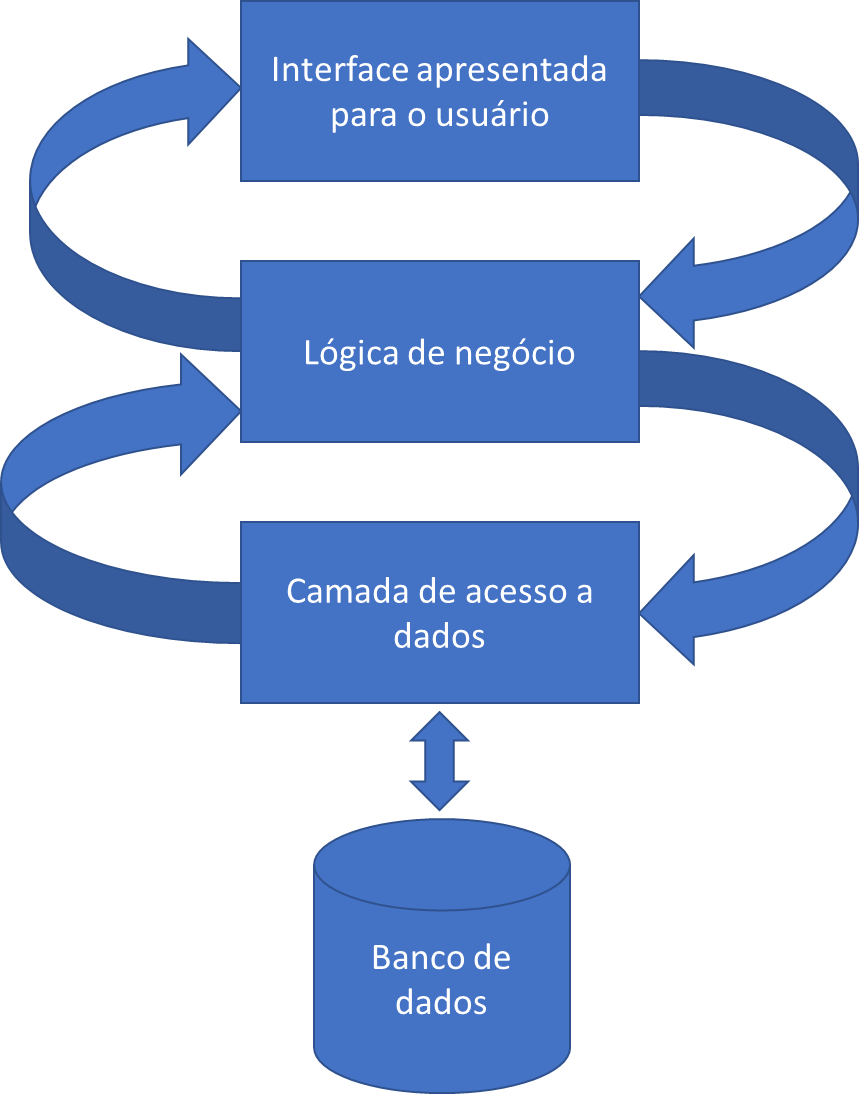
Após uma tentativa de registro, o sistema analisa se as credenciais são válidas. Em caso positivo, é gerado o e-mail com link de confirmação e em caso negativo são capturadas as mensagens de erro, exibindo-as logo em seguida ao usuário na tela de registro.

### Login



Figure

Através do fluxo apresentado na *figure 3*, pode-se observar o que foi abordado no **Caso de uso 02 – Login**.

Após a tentativa de login, o sistema realiza uma busca no banco de dados para verificar se as credencias passadas já existem. Em caso positivo, o usuário é autenticado com essas credenciais e é redirecionado para a página principal da aplicação, caso contrário, o sistema retorna mensagem de erro na tela de login.

Figure

### Fluxo de comunicação entre camadas

Em ambos os casos anteriores, é enviado algum dado da interface para a parte lógica, podendo ser identificado algum erro já nessa camada, sendo a mesma apresentada ao usuário. No caso do Login, por exemplo esse erro pode vir a ser identificado também após a comunicação com o banco de dados, já que a camada lógica ao fazer a consulta buscando pelas credenciais passadas pela interface, pode receber a informação de que esse registro ainda não existe no banco de dados.

A estrutura de comunicação pode ser exemplificada pela *figure 4*.

Através da modularização dos serviços, há a garantia de maior segurança, pois com uma maior quantidade de camadas, menor fica a responsabilidade de cada uma, prevenindo erros, facilitando a manutenção e aumentando a segurança em termos tentativas de acesso indevido.

## Requisitos de software

### Escopo do projeto

#### Objetivo

Criar uma rede social que tem como seu público alvo usuários que possuem ou administram hortas.

#### Funcionalidades básicas do sistema

* Cadastro / login de usuários
* Edição de perfil
* Criação de conteúdo
* Visualização de conteúdo criado por terceiros
* Visualização do perfil de terceiros

#### Necessidades do sistema

* Estar hospedado na plataforma Azure
  + Em período de desenvolvimento, poderá ser hospedado de forma local
* Possuir design visualmente aceitável e responsivo
* Possuir segurança entre usuários (e.g.: um usuário não poderá editar o perfil de outro)

### Análise de requisitos

#### Requisitos funcionais

* Cadastro de usuários mediante validação de dados
* Envio de e-mail contendo link para confirmação de criação de perfil
* Tela de login
* Tela de edição de perfil, na qual é possível alterar imagens e nome de usuário
* Criação de conteúdo através da página de detalhes do próprio perfil
* Criação de conteúdo através da página inicial
* Visualização de conteúdo criado por terceiros
* Visualização do perfil de terceiros, sem possibilidade de edição e sem acesso às suas informações privadas (e.g.: e-mail)

#### Requisitos não funcionais

* Banco de dados para salvar informações
* Conexão a um serviço externo para envio de e-mails
* Validações de campo (e.g.: e-mail, senha, confirmação de senha, campos vazios)
* Controladores para gerenciamento das ações recebidas pela URL
* Bloqueio de acesso a informações privadas de usuários que não o conectado

#### Requisitos do sistema

##### Detalhamento dos requisitos funcionais

* Cadastro de usuários mediante validação de dados
  + O sistema deverá receber os valores dos campos de cadastro após o usuário efetuar o *submit* do formulário. Após receber esses valores, o sistema deverá analisar se os mesmos são válidos ou não e em caso de erro, retornar uma resposta instrutiva e amigável ao usuário, a fim de que o erro seja corrigido
* Envio de e-mail contendo link para confirmação de criação de perfil
  + Deverá ser configurada uma API para o envio de e-mails. Através dessa API, será feita também a configuração do e-mail a ser enviado em caso de sucesso na criação de cadastro do usuário, e-mail qual irá redirecionar o usuário para uma tela de aviso, dizendo que a conta foi confirmada com sucesso, e em seguida o mesmo deverá ser redirecionado para a tela de login
* Tela de login
  + Deverá ser checado pelo sistema se o usuário está tentando acessar a rede social utilizando credenciais válidas ou não. Em caso positivo, o mesmo deverá ter seu acesso permitido, porém em caso negativo deve-se enviar uma mensagem de erro informando que as credenciais disponibilizadas não são válidas
* Tela de edição de perfil, na qual é possível alterar imagens e nome de usuário
  + Deverão ser persistidas as informações inseridas pelo usuário (nome de usuário, fotos de capa e perfil e o gênero). Informações de cadastro (e-mail e data de criação do perfil) não serão alteráveis
* Criação de conteúdo através da página de detalhes do próprio perfil
  + Primeiro, deverá ser identificado se o usuário está acessando a página de detalhes do seu próprio perfil ou não. Em caso positivo, será disponibilizado o botão “Editar”, redirecionando o mesmo para a tela de edição do seu perfil e também será disponibilizado o campo para criar novas postagens. Caso não seja o perfil do usuário conectado, campos como os botões mencionados anteriormente e também informações privadas como e-mail de cadastro, não deverão ser apresentados
* Criação de conteúdo através da página inicial
  + Na página inicial será possível criar novas postagens, assim como é possível na página de perfil do próprio usuário. Ao criar uma nova postagem, a página é atualizada contemplando também a última postagem realizada.
* Visualização de conteúdo criado por terceiros
  + Na página inicial serão apresentados de forma cronológica, ordenando pelo mais recente, as postagens feitas na rede social. As postagens poderão ser contempladas por texto, imagens ou ambos, mas não poderá ser criada uma postagem sem conteúdo algum. Cada postagem deverá informar quem foi seu criador, o conteúdo postado, data de criação e também terá links de redirecionamento para o perfil do criador da mesma.
* Visualização do perfil de terceiros, sem possibilidade de edição e sem acesso às suas informações privadas (e.g.: e-mail)
  + Um usuário poderá ter acesso ao perfil de outros usuários, porém sem qualquer permissão para criação de conteúdo no mesmo. Na página de perfil de cada usuário também estará visível o conteúdo (postagens) criado pelo mesmo

##### Detalhamento dos requisitos não funcionais

* Banco de dados para salvar informações
  + O banco deverá ter tabelas informando, essencialmente: contas de usuário para acesso (credenciais de login), contas de usuário da rede (perfil vinculado à credencial) e postagens realizadas
* Conexão a um serviço externo para envio de e-mails
  + Deverá ser configurada uma API para o envio de e-mails. Através dessa API, será feita também a configuração do e-mail a ser enviado em caso de sucesso na criação de cadastro do usuário, e-mail qual irá redirecionar o usuário para o sistema atribuindo o status de e-mail confirmado ao seu perfil no banco de dados
* Validações de campo (e.g.: e-mail, senha, confirmação de senha, campos vazios)
  + Deverão ser criados parâmetros para a criação de cada campo, sendo alinhado quais será o formado necessário para a senha, se qualquer e-mail (domínio público e privado) poderá ser aceito e quais campos poderão ser deixados em branco (perfil de usuário)
* Controladores para gerenciamento das ações recebidas pela URL
  + Através dos verbos HTTP recebidos pela URL, deverá ser administrado internamente quais serão suas funções e possíveis tratamentos de erro
* Bloqueio de acesso a informações privadas de usuários que não o conectado
  + Através da análise entre qual usuário está conectado, e qual perfil o mesmo deseja acessar, deve-se filtrar quais as informações serão exibidas

# Refinamento estrutural

## Identificação dos atributos das classes

Para o desenvolvimento do projeto, há a necessidade de três classes, que são respectivas ao perfil do usuário logado, perfil social e postagens realizadas.

### ProfileModel – A classe do perfil social

public class ProfileModel

{

[Key]

public Guid ProfileId { get; set; }

public string ProfileLoginId { get; set; }

public string ProfileUserName { get; set; }

public string ProfilePicture { get; set; }

public string ProfileBackgroundPicture { get; set; }

public string ProfileGender { get; set; }

public DateTime ProfileCreationTime { get; set; }

public virtual ICollection<PostModel> ProfilePosts { get; set; }

public ProfileModel()

{

this.ProfilePosts = new List<PostModel>();

}

}

#### ProfileId

Identificador único gerado para cada perfil.

#### ProfileLoginId

Identificador herdado do usuário logado.

#### ProfileUserName

Nome de perfil do usuário.

#### ProfilePicture

Foto de perfil do usuário.

#### ProfileBackgroundPicture

Foto de perfil do usuário.

#### ProfileGender

Gênero do usuário

#### ProfileCreationTime

Data e hora da criação do perfil.

#### ProfilePosts

Lista de objetos da classe *PostModel*. Nesse caso, indica que a classe *ProfileModel* possui uma relação de 1 x N em relação às postagens.

### PostModel – A classe das postagens

public class PostModel

{

[Key]

public Guid PostId { get; set; }

public DateTime PostCreationTime { get; set; }

public string PostCreator { get; set; }

public string PostDetails { get; set; }

public string PostPicture { get; set; }

}

#### PostId

Identificador único gerado para cada perfil.

#### PostCreationTime

Data e hora da criação do post.

#### PostCreator

Identificador herdado do perfil que criou a postagem.

#### PostDetails

Conteúdo textual da postagem.

#### PostPicture

Imagem inserida na postagem.

### AppContext – Code first

public class AppContext : DbContext

{

public DbSet<ProfileModel> AccountProfileModel { get; set; }

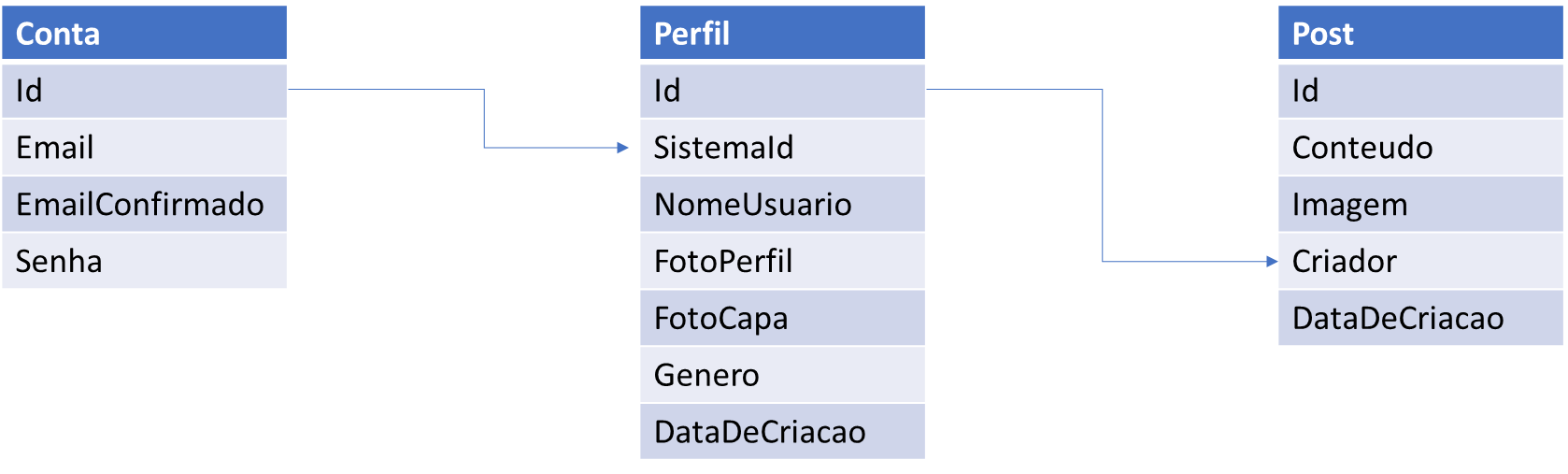
public DbSet<PostModel> PostModel { get; set; }

}

Ao utilizar o Entity Framework com Code First, há a necessidade da utilização do DbContext, o qual irá informar ao Framework quais classes devem ser criadas no banco de dados e quais as relações que irão possuir.

## Banco de dados

### Tabelas



Figure

### Conta

#### Id

Possui um identificador único nessa tabela, gerado automaticamente durante a criação da conta.

#### Email

Possui o e-mail fornecido durante a criação da conta.

#### EmailConfirmado

Possui um valor booleano informando se o e-mail foi confirmado ou não (se o link enviado para esse respectivo e-mail foi acessado ou não).

#### Senha

Possui a senha informada durante a criação da conta. O valor exibido no banco de dados é criptografado.

### Perfil

#### Id

Possui um identificador único nessa tabela, gerado automaticamente após a criação da conta.

#### SistemaId

Possui o identificador herdado de “Conta”. Essa informação é armazenada para identificar qual perfil é respectivo a cada conta. A relação entre Conta x Perfil é 1 x 1.

#### NomeUsuario

Nome fornecido durante a edição do perfil.

#### FotoPerfil

Imagem fornecida durante a edição do perfil.

#### FotoCapa

Imagem fornecida durante a edição do perfil.

#### Genero

Gênero informado durante a edição do perfil.

#### DataDeCriacao

Data e hora de criação do perfil.

### Post

#### Id

Possui um identificador único nessa tabela, gerado automaticamente após a criação da postagem.

#### Conteudo

Possui o conteúdo inserido no ato da postagem, caso exista.

#### Imagem

Possui a imagem inserida no ato da postagem, caso exista.

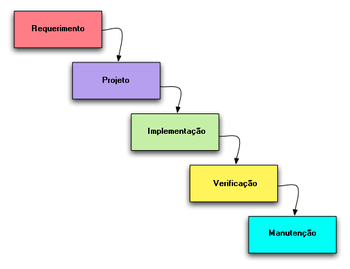
#### Criador

Id do perfil que realizou a postagem.

#### DataDeCriacao

Data de hora da criação da postagem

# Descrição Geral da estrutura

No modelo em cascata, as seguintes fases são seguidas em perfeita ordem:

* Requerimento
* Projeto
* Implementação
* Integração
* Teste e depuração (verificação)
* Manutenção de software

Seguindo esse padrão, foram feitos os requisitos funcionais e não funcionais do projeto, montado especificações de como deveria ser estruturado e desenvolvido de acordo com a necessidade do cliente/usuário.

Feito isso, começou o desenvolvimento em cima dessas especificações, levando em consideração as informações coletadas na primeira etapa do desenvolvimento.

Algo que ocorre nesse modelo em cascata no qual se sofre muito e a verificação do projeto com o cliente final, falaremos disso mais adiante.

Com o desenvolvimento do projeto, como seriam montados os bancos de dados (DB), camadas estruturais, desenvolvimento do código e hospedagem, foi realizada a etapa de implementação do projeto na Azure, onde tanto o web API, quando o projeto WEB foram configurados e deployados em uma URL na plataforma escolhida.

Problemas que foram encontrados por conta desta metodologia, são:

1. 1.Projetos reais raramente seguem o fluxo sequencial proposto pelo modelo. Embora o modelo linear possa conter iterações, ele o faz indiretamente. Como consequência, mudanças podem provocar confusão à medida que a equipe de projeto prossegue.
2. 2.Frequentemente, é difícil para o cliente estabelecer explicitamente todas as necessidades. O modelo cascata exige isso e tem dificuldade para adequar a incerteza natural existente no início de muitos projetos.
3. O cliente deve ter paciência. Uma versão operacional do(s) programa(s) não estará disponível antes de estarmos próximos ao final do projeto. Um erro grave, se não detectado até o programa operacional ser revisto, pode ser desastroso.

Ou seja, no modelo em cascata, qualquer alteração que o cliente precisa realizar, precisa esperar a etapa de validação, o que não ocorreria com a metodologia ágil e os MVPs, as pequenas entregas de valor, nesse modelo, enfrentamos o problema de, como decorrente do modelo “engessado” temos poucas interações, cliente X fornecedor, com isso, o cliente só ira verificar o andamento do desenvolvimento quando o software estiver na etapa de verificação, o que na maioria das vezes acarreta alterações de layout, escopo e até mesmo variações de regras de negócios. O que acaba muitas vezes levando o projeto a voltar para a etapa de Requerimento do usuário para pegar novas informações e assim, dificultando o desenvolvimento da aplicação.

Hoje, trabalhando com metodologias ágeis, já enfrentamos problemas da equipe de produto pedir alterações de escopo no decorrer de uma task, isso no “mundo real”.

Levando em consideração o projeto em modelo cascata, a maior dificuldade são as alterações de desenvolvimento seja por problemas na captação de informações, seja por alteração de escopo, problemas na arquitetura, que todos esses vão acabar sendo vistos somente na etapa de verificação.

Levando em conta um projeto de conclusão de tempo médio de 6 meses, levantando um custo médio de 20/h semanais ou 480h nos 6 meses. Se pararmos para verificara s vezes que o projeto precisou voltar para a etapa de requerimento por falta de informações, conhecimento ou mudança de ideias, essas 480h podem chegar ao dobro do tempo, o que se fossemos colocar em custos de projeto seriam completamente, “insanos”.

Modelos mais flexíveis e com maior interação cliente X desenvolvedor são mais usuais e poderiam evitar esses contratempos de montagem de projeto, além de muitas vezes, entregar o projeto em menor tempo e com “mais valor atribuído”.

Para esse projeto, diria que seria mais pratico e eficiente a utilização de pequenas entregas de valor, menos documentações extensas, utilização do próprio código como uma fonte de documentação e assim otimizando e agilizando o tempo do desenvolvedor.

Por fim, após passarmos pela etapa de verificação, entramos na etapa de manutenção do código, onde vamos resolver alguns bugs que apareceram após a validação, correção de possíveis problemas com a plataforma de deploy (azure), dentre outros problemas que possam ocorrer.