MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
Curso de Sistemas de Informação
Prof. Francisco das Chagas Imperes Filho
Alunos: Diego Fernando de Sousa Lima
Antonio Ferreira de Carvalho

FASE IV

DETALHAMENTO DO BANCO DE DADOS E TESTE DE USABILIDADE APLICADO A PLATAFORMA *FIXCODE*

PICOS
2017

1. INTRODUÇÃO

FixCode é uma plataforma desenvolvida pensada nos desenvolvedores de softwares. O intuito inicial é a interação entre os vários desenvolvedores e programadores que existem mundo afora, e para isso a comunidade leva consigo a ideia do auxílio aos programadores. O projeto está parcialmente concluído em (https://fixcode.herokuapp.com) e é livre no repositório (https://github.com/diegofsousa/FixCodeProject).

Este documento trata do detalhamento do banco de dados e teste unitário dos princípios de IHC realizado com base na ferramenta **ErgoList** (http://www.labiutil.inf.ufsc.br/ergolist/).

2. DETALHAMENTO DO ESQUEMA DE DADOS

Ao todo, o esquema de banco de dados da plataforma **FixCode** tem 26 entidades somando as escritas e migradas através do arquivo *models.py* como também as pré-definidas pela framework **Django**.

A representação gráfica do modelo relacional está no anexo que vai junto a este documento (banco.png).

Dentre as tabelas da aplicação, as que estão em maior relevância são:

- `auth_user`;
- `core_profile`;
- `core_followers`;
- `core_fixies`;
- `core_comentfixies`;
- `core_post`
- `core_comentpost`;
- `core_participations`;
- `core_favorites`;
- `core_areas`;
- `core_message`;
- `core_blocked`.

2.1. USUÁRIOS E PERFIS

As tabelas **`auth_user`** e **`core_profile`** estão estritamente relacionadas pela relação *OneToOne*. A entidade `core_profile` funciona como uma extensão de `auth_user` e isso acontece por que `auth_user` é uma entidade gerada pela framework Django já com seus atributos definidos e de difícil alteração. Já a `core_profile` foram adicionados novos atributos para o usuário:

- `id`;
- `bio` (Breve descrição para o usuário);
- `git` (Link para perfil do *GitHub)*;
- `imagem_perfil` (Diretório para a imagem de perfil);
- `data_cadastro` (Tempo exato em que o usuário de cadastra no sistema);
- `visto_por_último` (Tempo exato em que usuário usou a aplicação por último);
- `ativo` (Verificação de aptidão para uso da plataforma);
- `expert` (Verificação de primeiro acesso do usuário);
- `user_id` (Chave estrangeira de `auth_user`).

2.2. POLÍTICA DE SEGUIDORES

É adotado na plataforma FixCode um esquema de seguidores, semelhante ao implantado na rede social *Twitter*. Neste caso, um cada usuário pode seguir e ser seguido por várias pessoas, constituindo assim, uma relação *ManyToMany*.

A entidade responsável pela relação entre os usuários é a **`core_followers`**. Esta armazena, além chave estrangeira dos dois usuários, um campo para a data de quando aconteceu o evento.

2.3. FIXIES E POSTS

Fixies (`core_fixies`) e ***Posts*** (`core_post`) são abstrações para, respectivamente, questionamento e partilha de conhecimento.

Os atributos da entidade `core_fixies` são:

- `id`;
- `titulo`;
- `descrição`;
- `data`;
- `resolvido` (Verificação de quando o criador não o considera como resolvido);
- `tem_melhor_resposta` (Verificação de quando o criador seleciona uma melhor resposta);
- `notificacao` (ver item 2.4);
- `ativa_notificacao` (ver item 2.4);
- `user_id` (Chave estrangeira de `auth_user`).

Os atributos da entidade `core_post` são:

- `id`;
- `titulo`;
- `post` (Texto da postagem);
- `data`;
- `notificacao` (ver item 2.4);
- `ativa_notificacao` (ver item 2.4);
- `exibir_perfil` (Verificação de quando o criador desejar que o post seja exibido no perfil);
- `anexo` (Diretório para a arquivo enviado junto a postagem);
- `user_id` (Chave estrangeira de `auth_user`).

2.3.1 RESPOSTAS DE FIXIES E COMENTÁRIOS EM POSTS

Cada *Fix* pode ter várias repostas, então há uma relação *OneToMany* entre `auth_user` e **`core_comentfixies`**, que é a entidade responsável pelas repostas dos *fixies*. Os atributos de `core_comentfixies` são:

- `id`;
- `coment` (Texto da resposta);
- `data`;
- `melhor_resposta` (Verificação para saber se o criador do *fix* classificou esta resposta como sendo a melhor);
- `fixie_id` (Chave estrangeira de `core_fixies`);

- `user_id` (Chave estrangeira de `auth_user`).

Assim como nos *fixies*, os posts também tem uma entidade que fica responsável pelo *feedback* de cada post, esta tem o nome de **`core_comentpost`** e tem menos atributos que `core_comentfixies`:

- `id`;
- `coment` (Texto do comentário);
- `data`;
- `post_id` (Chave estrangeira de `core_post`);
- `user_id` (Chave estrangeira de `auth_user`).

2.3.2. PARTICIPAÇÕES E FAVORITOS

Toda vez que um usuário comenta um *fix*, criará um registro na tabela **`core_participations`**. Por sua vez, `core_participations` tem a seguinte estrutura:

- `id`;
- `notificacao` (Quantas respostas foram dadas no mesmo *fix* após o usuário ter comentado);
- `ativa_notificacao` (Verificação se usuário deseja ser notificado pelo *fix*);
- `fixie_id` (Chave estrangeira de `core_fixies`);
- `user_id` (Chave estrangeira de `auth_user`).

Há também a possibilidade de um usuário favoritar um *fix* através de um botão na página. A cada vez que um usuário favoritar um fix um novo registro na tabela **`core_favorites`** é criado. Os atributos de `core_favorites` são os mesmos de `core_participations`.

Ambas, as entidades `core_participations` e `core_favorites` servem justamente para o controle de notificações da plataforma. Mais detalhes no item 2.4.

2.4. NOTIFICAÇÕES PARA OS USUÁRIOS

A possibilidade de notificações se dá para o usuário criador pelos atributos `notificacao` das entidades `core_fixies` e `core_post`. Quando esses atributos têm valores maiores que 0 (zero), a aplicação, quando consultada, devolve a mensagem para o autor alertando-o sobre o *fix* ou *post* notificado.

Outra maneira de notificar o usuário se dá quando ele participa ou favorita algum *fix*. Neste caso, quem se armazena o valor-chave das notificações é o atributo `notificacao` de `core_participations` e `core_favorites`.

2.5. ÁREAS/HABILIDADES E SUA IMPORTÂNCIA

As entidades `core_profile`, `core_fixies` e `core_post` tem um relacionamento ManyToMany com core_areas. Acontece que, boa parte do que o usuário vê na página principal se dá pelo conjunto de afinidades escolhidas (**`core_areas`**), por exemplo, o post que tem as areas *'C'*, *'Python'* e *'Arduíno'* será exibido na *timeline* do usuário que escolheu *'Java'*, *'Android'* e *'Arduíno'* justamente pela habilidade em comum.

2.6. SISTEMA DE MENSAGENS INBOX

A entidade **`core_message`** é responsável por armazenar mensagens de um usuário ao outro. Sua estrutura detalhada é:

- `id`;
- `data` (Tempo em que a mensagem foi enviada);
- `texto` (Conteúdo da mensagem);
- `visualisada` (Verificação para se a mensagem foi vista pelo destinatário);
- `emissor_id` (Chave estrangeira do `auth_user` que emitiu a mensagem);
- `receptor_id` (Chave estrangeira do `auth_user` que recebeu a mensagem).

`core_message` recebe chave estrangeira de dois usuários distintos. Na aplicação, é feito uma seleção entre as mensagens que estão relacionadas entre estes dois usuários, mas isso já é outra história.

2.7 SISTEMA DE BLOQUEIO DE USUÁRIOS

Por questão de privacidade e segurança do usuário, também foi desenvolvida uma entidade para bloqueio de usuários chamada **`core_blocked`**. Funciona assim: qualquer usuário poderá bloquear qualquer usuário e após feita esta ação, o usuário bloqueado não poderá interagir com o usuário que o bloqueou em seus *fixies*, *posts* e mensagens inbox.
A estrutura de `core_blocked` foi montada assim:

- `id`;
- `data` (Tempo em que houve o bloqueio);
- `user_id` (Chave estrangeira do `auth_user` realiza o bloqueio);
- `userblocked_id` (Chave estrangeira do `auth_user` bloqueado).

2. TESTE DE USABILIDADE