Apresentação do Web APP “Próximo”

APLICATIVO PARA AGILIZAR PEDIDOS DE REFEIÇÕES

Arthur Barcellos[1], Caroline Uchoa[1] e Gabriel Bastos[1]

[1]Departamento de Engenharia Eletrônica – Universidade Federal do Rio de Janeiro (UFRJ)

arthurbarcellos@poli.ufrj.br, carol_uchoa8@poli.ufrj.br, gabriel.bastos@poli.ufrj.br

Resumo. *Criamos um web app direcionado a agilizar a o pedido e a preparação de comidas para os restaurantes do CT (centro de tecnologia), onde através de uma pagina web, o cliente pode ver o cardápio de vários restaurantes e fazer seu pedido via internet para as filas se tornarem menores, e o almoço mais rápido.*

1. Motivação

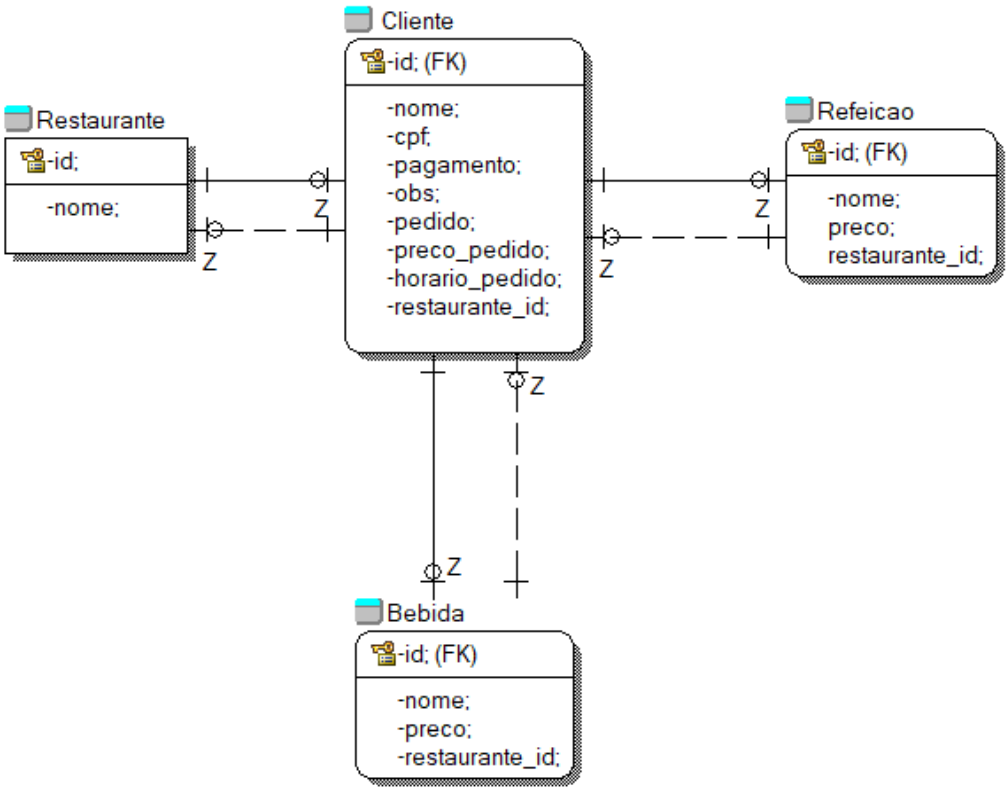
É notável a quantidade de problemas existentes no dia-a-dia brasileiro. Tais como filas, burocracia, ineficiência de estocagem, entre outros. Com a intenção de amenizar um problema cotidiano, resolvemos trabalhar em um aplicativo visando a agilidade no processo de compra de refeições, utilizando orientação a objetos em Python em conjunto com um banco de dados criado no MySQL.

2. OBJETIVO

Tendo em vista as extensas filas que nos cercam recorrentemente, propusemo-nos a criar um aplicativo, chamado Próximo, para reduzir este problema. Este consiste basicamente em um processo de autoatendimento para restaurantes e praças de alimentação. Onde o cliente, ao chegar no local desejado, realiza seu pedido pelo celular. Tudo sem precisar sair da mesa. O restaurante receberá esta informação, produzirá a refeição, e anunciará ao cliente quando a mesma estiver disponível para retirada. O aplicativo, portanto, traz conforto ao cliente e ao mesmo tempo reduz gastos com funcionários para uma empresa.

3. Modelo de Entidade e Relacionamento

O modelo abaixo ajuda a entender melhor como todo o programa vai funcionar e como as classes se intercomunicam e se relacionam.



4. Código fonte

Temos no código fonte a área de templates (arquivos em HTML) e a área onde estão os arquivos .py. Como os templates forão usados de um site, deixaremos na bibliografia onde pegamos.

Falaremos um pouco sobre os templates e o que eles mostram:

Na página "Home": Temos a chamada da tabela de restaurantes que estão cadastrados no banco de dados, onde o usuário vai selecionar um dos restaurantes da lista. Logo depois, teremos uma chamada da tabela das refeições do restaurante que foi selecionado para se selecionar um dos pratos feitos. Feito isso, termos a parte das bebidas onde tambem é chamada a tabela das bebidas do restaurante especificado.
Com tudo selecionado, o usuario faz o cadastro de seus dados, como nome, CPF, forma de pagamento e um "Obs" para finalizar o pedido, e vizualizar tudo o que foi pedido e confirmar.

Na aba "Ver pedidos": Nessa aba, acessamos a página onde é possível selecionar um restaurante e ver a lista de pedidos na ordem, para consultar quantos pedidos há na sua frente ou apenas ver a lista de pedidos do restaurante selecionado.

Na aba "Login" : Essa aba foi feita para os administradores poderem acrescentar novos restaurantes, pratos e bebidas no banco de dados, e assim ter uma inserção de dados mais fácil rápida do sistema.

Falaremos dos arquivos .py, que é onde a mágica acontece.

- Flask_app:
  O Flask é onde ocorre a interseção entre os dados e o Framework. Nele temos o flask recebendo o banco de dados que será usado para realizar as consultas.
  Primeiro, vamos associar ao flask, o banco de dados que será utilizado para pegar as insformações que serão mostradas na tela pelas páginas. Depois, usaremos os decoradoes para associar funções as rotas postas no decorador, para assim ele chamar os templates desejados e mandar as informações corretadas do banco de dados para seus respectivos templates.
  Os decoradores, por sua maioria, são para direcionar para páginas da leitura do banco de dados, mas a página após o pedido ser realizado e confirmado, terá todos os dados que foram preenchidos pelo usuario sendo cadastrados na tabela "cliente" para que possa ser visualizado na página "Ver pedidos".

- Classes.py:
  Como o nome ja diz, é o arquivo que contem as classes que vão ser chamadas no ‘flask_app’, onde cada classe lê seuas respectivas tabelas e associa seus valores a strings para serem utilizadas no ‘flask_app’.

- Lojas.py:

  Esse arquivo faz a inserção dos dados necessários dentro da tabela, para que eles sejam utilizados no ‘classes.py’ e consequentemente no ‘flask_app.py’.

5. Bibliografia

- BORGES, Luiz Eduardo. **Python para Desenvolvedores**. 3 ed. São Paulo. Novatec. 2014

- POLEPEDDI, Lalith. **Uma Introdução a Python Flask Framework.** Envato Tuts +. Disponível em: https://tutsplus.com. Acesso em: 21/10/2018