UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES CIÊNCIAS E HUMANIDADES
EACH - USP

RELATÓRIO SOBRE O TRABALHO

DISCIPLINA: SOLUÇÕES WEB BASEADAS EM SOFTWARE LIVRE
DOCENTE: MÁRCIO RIBEIRO
ALUNO: BRUNO YOSHIKAZU SHIMADA
N.USP: 7136630
ALUNO: VICTOR NASCIMENTO KULCSAR
N.USP: 7136432
TURMA 02

1. INTRODUÇÃO

Para este trabalho foi pedido que fosse desenvolvido um sistema web, ou seja que possibilita acesso remoto a aplicação, em linguagem livre e que tenha interação com um banco de dados. Aproveitando das aulas assistidas durante o semestre, e segundo também a recomendação da especificação, o trabalho foi desenvolvido na linguagem Ruby usando o framework Rails.

Locadoras de videogame eram extremamente populares a alguns anos atrás, até o surgimento do Playstation e o uso de uma mídia barata e que era facilmente pirateada, o CD.

Eram locais em que se podiam alugar jogos ou até mesmo jogar no local, mas o foco e a principal atividade era a primeira.

Atualmente é difícil imaginar que uma locadora seja um negócio viável para se manter, mas para o trabalho foi imaginado que hoje em dia esse é um bom modelo de negócio.

O objetivo do trabalho foi desenvolver um sistema simples que simula um gerenciamento simplificado de uma locadora de jogos de videogame.

As funcionalidades básicas necessárias para se gerenciar uma locadora, seriam funcionalidades que cuidam dos clientes do negócio, dos produtos disponíveis, e o gerenciamento dos empréstimos, ou alugueis que são realizados.

2. METODOLOGIA

Para se desenvolver e testar o trabalho foram usados os seguintes items:

- Ferramentas

    - Atom - Versão 0.149.0

    Editor de texto desenvolvido pela equipe do GitHub. Leve, rápido, open source e altamente customizável o que faz dele um ótimo editor para se trabalhar com as mais variadas linguagens. Foi usado para escrever todos os códigos da aplicação.

- pgAdmin - Versão 1.18.1

Ferramenta open source que prove uma interface gráfica para se adiministrar o banco de dados PostgreSQL. Foi usado para se gerenciar as tabelas e os dados criados pela aplicação

- GitHub for Windows - Versão Great Dane(2.6.6.2) ea1880c

Versão para desktop do GitHub, fornece além de um shell para se trabalhar via linha de comando, uma ferramenta visual para facilitar o trabalho. Foi usado para controle de versão e entrega do trabalho.

- DB-Main - Versão 9.2.0

Ferramenta para criação de diversos tipos de diagramas. Foi usado para a elaboração do modelo entidade-relacionamento da aplicação.

- Google Chrome - Versão 39.0.2171.95.m

Navegador usado para se testar a aplicação.

- Command Prompt

Usado para execução dos comandos do Rails.

❖ Linguagens

- Ruby - Versão 1.9.3p551

Linguagem usada na aplicação, código colocado em diversos pontos da aplicação, por exemplo nos controladores.
A versão das rubygems usada é a 2.3.0, não foi usada a mais atual porque ela dava um erro que não foi achado uma maneira de contornar o mesmo. Com a versão mais nova do pacote um erro ocorria ao se tentar criar uma aplicação usando outro banco de dados sem ser o SQLite3, por mais que se forçasse a instalar a gem para trabalhar com PostgreSQL ou MySQL o erro persistia.
A versão do DevKit usada foi a tdm-32-4.5.2. o DevKit é uma ferramenta que facilita o uso de extensões nativas em linguagem C/C++ para o Ruby no Windows.

- HTML/CSS

Foi usado na parte da criação das Views da aplicação. Apesar de serem geradas automaticamente pelo Rails, foi sentido a necessidade de se alterar um ou outro pedaço no código da View para deixar ela da maneira desejada.

- Banco de Dados

    - PostgreSQL - Versão 9.3.5

  Banco de fácil uso e que foi o que melhor integrou com o Rails na nossa opnião.

- Framework

    - Rails - Versão 4.2.0
    - Um framework para aplicações web open source escrito em Ruby. É um framework que enfatiza o uso de paradigmas e padrões conhecidos de engenharia de software como o Convenção Acima de Configuração (convention over configuration / CoC), o não se repita (don't repeat yourself / DRY) e o Model-View-Controller (MVC).
    - O MVC talvez seja o ponto mais importante do framework Rails, foi percebido que a aplicação toda é baseada nesse padrão. Ele consiste em construir a aplicação em 3 camadas. O Model, falando de modo simples, é onde ficam os arquivos relacionados as entidades, as tabelas. A View são os arquivos que interagem com o usuário, são as telas da aplicação. O Controller são os arquivos que gerenciam as entidades, cada tabela posssui seu controller, e nesses controllers estão as ações CRUD (Create,Read,Update,Delete).

Sobre o desenvolvimento:

O projeto teve duas tentativas de ser realizado. Na primeira tentamos fazer manualmente o projeto, criando cada parte do MVC individualmente e configurando manualmente as rotas da aplicação. Apesar de ser uma maneira melhor de se entender a estrutura do projeto, demorou muito tempo até ele ser construído e infelizmente o mesmo não funcionava, dando constantemente o erro de código 500, que por falta de logs mais explicítos dificultavam o entendimento do erro.

Estudando um pouco mais e observando alguns tutorias descobrimos uma maneira mais rápida, eficiente e que faz uso dos comandos disponíveis por meio do Rails. Essa maneira era o uso do comando Scaffold, passando por paramêtro os atributos desejados que uma entidade num banco de dados possua, este comando criava automaticamente todas as camadas da aplicação relacionadas a essa tabela e

configurava suas rotas, o que facilitou muito e acelerou a criação e finalização do projeto.

O caminho seguido para se criar a aplicação foi:

1. Dentro de um diretório criar o projeto rails usando :
   rails new [nome-projeto] -d postgresql
2. Dentro do diretório do projeto, criou-se os banco de dados que seriam usados pela aplicação:
   rake db:create

Este comando criava três bancos diferentes a ser usados dependo da fase do projeto, o banco para o desenvolvimento, o de teste e o de produção

3. Nesse passo eram usados dois comandos juntos, no primeiro usava se o scaffold para criar todos os arquivos relacionados a tabela passada no scaffold, nisso fizemos as alterações que julgamos necessárias e depois outro comando para de fato criar as entidades no banco de dados. Os comandos desse passo são:
   rails generate scaffold [nome-modelo] [nome-atributo:tipo]...
   rake db:migrate
4. O passo 3 foi executado 3 vezes na nossa aplicação para gerar os 3 modelos necessários, vale ressaltar que na execução do scaffold as rotas do projeto eram atualizadas automaticamente.
5. Fizemos as alterações que julgamos necessárias nas camadas de controller e view e por fim iniciava-se o servidor para testar e ver a aplicação rodando. O comando foi:
   rails server

Os testes feitos foram apenas para se verificar se as operações CRUD estavam sendo propriamente executadas, e se o modelo que representava o relacionamento estava funcionando como deveria.

Para se testar o projeto em outra máquina é necessário ter instalado no mínimo o Ruby, a gem do framework Rails e o PostgreSQL.

Depois de clonado ou baixado o master do projeto no GitHub, acessar via linha de comando o diretório nomeado ‘locadora’.

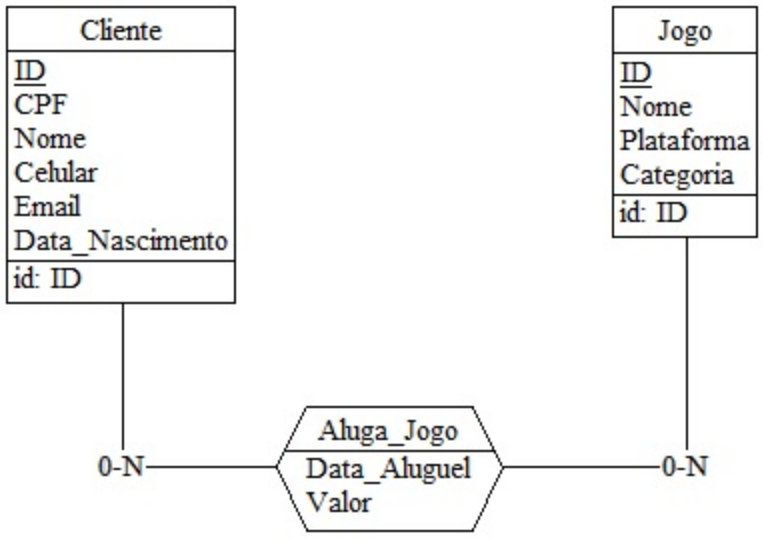
A seguir é um ponto que gerou dúvida já que a aplicação foi desenvolvida em uma única máquina. Assumimos que devem ser executados os comandos de criar os bancos (rake db:create) e de migração dos modelos (rake db:migrate).

Então finalmente é só iniciar o servidor usando o comando

rails server

Acreditamos que o banco ira não populado, devendo então ser populado para se poder testar a aplicação. Um arquivo de texto contendo as especificações do modelo será disponibilizado.

O diagrama Entidade-Relacionamento:



3. RESULTADOS

Foi implementado os modelos apresentados acima no diagrama com todos funcionando e executando todas as operações que são esperadas que uma aplicação com acesso a banco de dados seja capaz de fazer.
O que faltou ser feito foram algumas melhorias no modelo do relacionamento para que na exibição de detalhes (*action 'show'*) não se mostrasse só o número do id do cliente ou do jogo, mas sim o seu nome.

4. CONCLUSÃO

O framework Rails é uma ferramenta muito poderosa para desenvolvedores que criam aplicações direcionadas a web, facilitando muito o trabalho que talvez fosse gasto antes do desenvolvimento dele, acelerando também esse processo, como mostra um famoso exemplo na internet que ensina a montar um blog funcional usando Rails em menos de 15 minutos.