BCC | Engenharia de Software

Projeto da Disciplina

Docente: Rodrigo Andrade

Organização: hotelaria-ufape

Membros: Guilherme Rutemberg | Gustavo Henrique | Mauro Vinícius

Max David | Pedro Augusto | Pedro Almeida

**Postmortem: Segunda Iteração**

a) Período:

- 11/07/2023
- 03/08/2023

b) O que estava planejado?

| Atividades para a Segunda Iteração | |
|---|---|
| **Tarefa** | **Responsável** |
| Criação do Scaffold de “Quarto” | Max David |
| Criação do Scaffold de “Cliente” | Max David |
| Criação do Scaffold de “Reserva” | Max David |
| Adição de Validações no Model “Quarto” | Guilherme Rutemberg |
| Adição de Validações no Model “Cliente” | Guilherme Rutemberg |
| Adição de Validações no Model “Reserva” | Guilherme Rutemberg |
| Adição de Busca Alternada no Controller “Reserva” | Max David |
| Adição de Busca Alternada no Controller “Cliente” | Mauro Vinícius |
| Adição de Busca Alternada no Controller “Quarto” | Mauro Vinícius |
| Criação do Controller, View e Rota para “Home” | Guilherme Rutemberg |
| Instalação e Execução do Capybara e Cucumber | Pedro Augusto |
| Criação de 5 Testes de Cenário com “Quarto” | Pedro Augusto |
| Criação de 5 Testes de Cenário com “Cliente” | Pedro Almeida |
| Criação de 5 Testes de Cenário com “Reserva” | Gustavo Henrique |
| Criação de 1 Teste Unitário para “Quarto” | Pedro Augusto |
| Criação de 1 Teste Unitário para “Cliente” | Pedro Almeida |
| Criação de 1 Teste Unitário para “Reserva” | Gustavo Henrique |
| Adição de Tradução e CSS para todas as Views | Mauro Vinícius |
| Publicação de uma nova Release | Max David |
| Deploy no Render | Max David |

c) O que foi feito?

| Atividades para a Segunda Iteração | |
|---|---|
| **Tarefa** | **Responsável** |
| Criação do Scaffold de “Quarto” | Max David |
| Criação do Scaffold de “Cliente” | Max David |
| Criação do Scaffold de “Reserva” | Max David |
| Adição de Validações no Model “Quarto” | Guilherme Rutemberg |
| Adição de Validações no Model “Cliente” | Guilherme Rutemberg |
| Adição de Validações no Model “Reserva” | Guilherme Rutemberg |
| Adição de Busca Alternada no Controller “Reserva” | Max David |
| Adição de Busca Alternada no Controller “Cliente” | Mauro Vinícius |
| Adição de Busca Alternada no Controller “Quarto” | Mauro Vinícius |
| Criação do Controller, View e Rota para “Home” | Guilherme Rutemberg |
| Instalação e Execução do Capybara e Cucumber | Pedro Augusto |
| Criação de 5 Testes de Cenário com “Quarto” | Pedro Augusto |
| Criação de 5 Testes de Cenário com “Cliente” | Pedro Almeida |
| Criação de 5 Testes de Cenário com “Reserva” | Gustavo Henrique |
| Criação de 1 Teste Unitário para “Quarto” | Pedro Augusto |
| Criação de 1 Teste Unitário para “Cliente” | Pedro Almeida |
| Criação de 1 Teste Unitário para “Reserva” | Gustavo Henrique |
| Adição de Tradução e CSS para todas as Views | Mauro Vinícius |
| Publicação de uma nova Release | Max David |
| Deploy no Render | Max David |

d) O que não foi feito?

- Tudo que foi requisitado para a segunda iteração foi feito com êxito.

e) O que está planejado para próxima iteração?

- Revisão do Código CSS para melhor estilização das Views;
- Novas Lógicas de Negócio, como, por exemplo: credenciais de acesso;
- Chance de migração para o Banco de Dados para o Render para consistência de dados pós-deploy.

f) Lições aprendidas

- Com o desenvolvimento contínuo do projeto, foram tomadas algumas práticas de construção de software pela equipe, sendo elas:

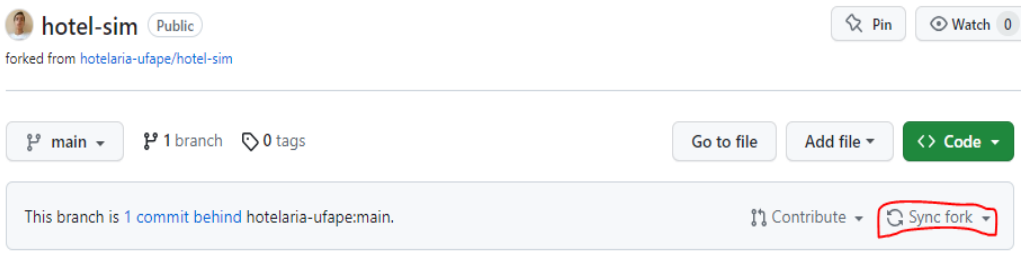
- Configurações de Repositório e Práticas de Sincronização;
- Configurações de Código e Indentação.

Veja mais sobre cada um dos processos adotados abaixo.

f.1) Configurações de Repositório e Práticas de Sincronização.

Com o ambiente de desenvolvimento devidamente configurado, houveram ressalvas sobre como agir conforme a manutenção do repositório principal e de suas variantes:

1. **Sempre**, ao iniciar uma rotina de trabalho com o projeto, acesse o seu repositório remoto dentro da plataforma Github, para que assim, visualize, caso haja, alguma desatualização do seu repositório remoto em relação ao repositório principal. Caso haja, aperte em “Sync fork”. Dentro da IDE, utilize a opção “update” para também atualizar seu repositório local.




2. **Sempre** que criar um novo commit para o projeto, para otimizar a busca por commits em histórico, lembre-se de seguir essas notações:

   2.1. Ao utilizar um comando Rails na plataforma para gerar algo automatizado, utilize a seguinte descrição para o commit:

   - **Inicializando o(a) “recurso” “x” através de “comando”.**
     (substitua “recurso” pelo que foi gerado: classe ou controlador)
     (substitua “x” pelo nome do recurso gerado)
     (substitua “comando” por: generate + scaffold, controller ou model, seguido dos atributos e seus tipos de dados)

   2.2. Ao adicionar um novo arquivo ao projeto que contenha algum propósito, utilize a seguinte descrição para o commit:

   - **Adicionando o(a) “recurso” para “propósito”.**
     (substitua “recurso” pelo nome e tipo do arquivo adicionado)
     (substitua “propósito” pelo objetivo de sua adição)

   2.3. Ao editar um arquivo existente no projeto para algum propósito, utilize a seguinte descrição para o commit:

   - **Atualizando o(a) “recurso” para “propósito”.**
     (substitua “recurso” pelo nome e tipo do arquivo atualizado)
     (substitua “propósito” pelo objetivo de sua modificação)

   2.4. Ao remover um arquivo existente no projeto para algum propósito, utilize a seguinte descrição para o commit:

   - **Removendo o(a) “recurso” para “propósito”.**
     (substitua “recurso” pelo nome e tipo do arquivo removido)
     (substitua “propósito” pelo objetivo de sua remoção)

b) Configurações de Código e Indentação:

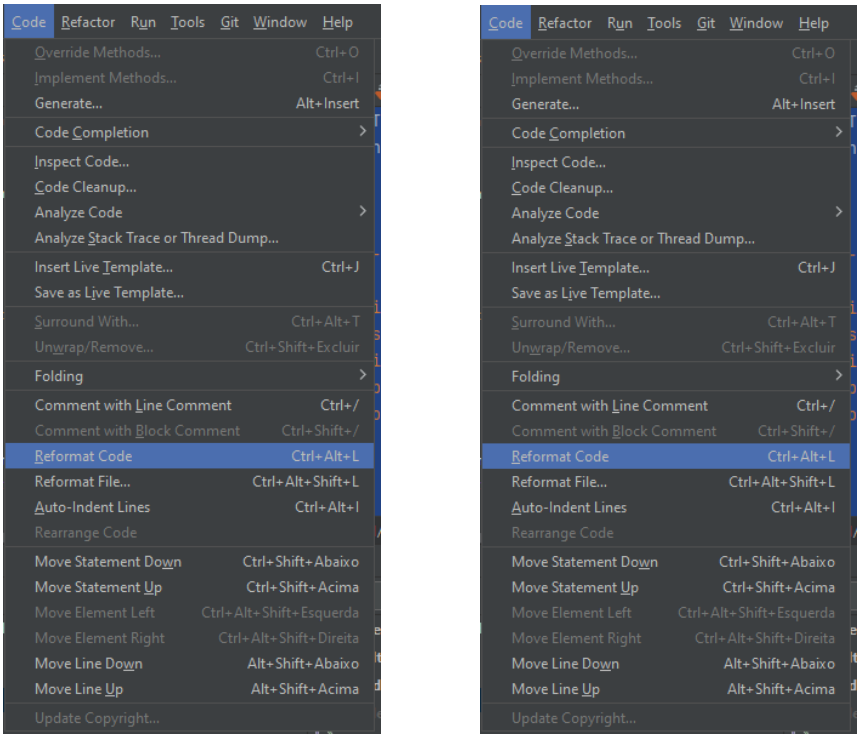
Seguindo a própria implementação de código que o Rails realiza ao gerar suas funcionalidades automaticamente, a convenção de código do projeto se baseará em:

1. **Variáveis e Métodos:** serão escritos com "snake_case" e não serão resumidos, ou seja, dado um método "Chamar funcionário", em código, obtém-se "chamar_funcionario".

2. **Classes e Módulos:** serão escritas com "CamelCase" e não serão resumidas, ou seja, dado uma classe "Funcionário gerente", em código, obtém-se "FuncionarioGerente".

3. **Tradução de Código:** serão implementados, dentro de todo o back-end e front-end do projeto, funcionalidades, testes e automatizações, através de termos e palavras em português.

4. **Indentação de Código:** será implementada ao final de toda edição de código através dos comandos, em caso de IDE RubyMine, control + a, seguido da seleção da opção "Reformat Code" da aba "Code".




Procedimento de indentação de código automatizado pelo RubyMine.