

Testes automatizados de software: Uma obrigação do programador

Palestra para o UN1D3V TECH DAY 2 - UNIME
Por: Pablo Ricardo Roxo Silva

Sobre o palestrante

- Adoro códigos bem escritos e com qualidade
- Estilo multidisciplinar na carreira
- Estudo programação desde 2001
- Engenheiro de Software Sênior na PRODEB
- Ex-professor da UNIME e do SENAI
- Pós-graduando em Arquitetura de Software
- Pós-graduado em Ciência de Dados & Big Data
- Pós-graduado em Dispositivos Móveis
- Bacharel em Ciência da Computação
- <https://linkedin.com/in/pabloroxo>

Agenda

- A quem se destina esta palestra
- O dia a dia do desenvolvedor
- Back-end com Laravel
- Testes automatizados com PHPUnit
- Cobertura de código com Xdebug
- Ao infinito e além

A quem se destina esta palestra

- Estudantes que têm pouca ou nenhuma vivência na área de desenvolvimento;
- Profissionais que atuam com quaisquer linguagens de programação e ferramentas;
- Professores(as) e pesquisadores(as) que querem revisar os conceitos teóricos e práticos apresentados;
- Empresários(as) que desejam modernizar processos de produção com o estado-da-arte;
- Todos(as) que querem agregar conhecimento.

O dia a dia do desenvolvedor

- Um time de desenvolvimento é formado por:
 - Um(a) **Scrum Master** (ou “agilista”);
 - Pode ser o(a) agilista de mais de um time.
 - Um(a) **Product Owner** (ou “P.O.”);
 - Pode ser o(a) P.O. de mais de um projeto.
 - Um(a) ou mais **engenheiros(as) de front-end**;
 - Um(a) ou mais **engenheiros(as) de back-end**;
 - Um(a) **analista de testes** (ou “Q.A.” - Quality Assurance).
- Observações:
 - O(a) **engenheiro(a) full-stack** atua tanto no front-end quanto no back-end e até em mais áreas.

O dia a dia do desenvolvedor

- Outros papéis importantes comuns aos times:
 - Um(a) ou mais administradores(as) de bancos de dados (ou “D.B.A.” - Database Administrator);
 - Um(a) líder técnico(a) (ou tech lead);
 - Um(a) DevOps (Development Operations)
 - Um(a) gerente de projetos (ou “P.M.” - Project Manager);
 - Entre outros papéis mais específicos.
- Observações:
 - Às vezes o(a) tech lead também é o(a) DevOps;
 - Existe ainda o(a) DevSecOps, que integra segurança da informação.

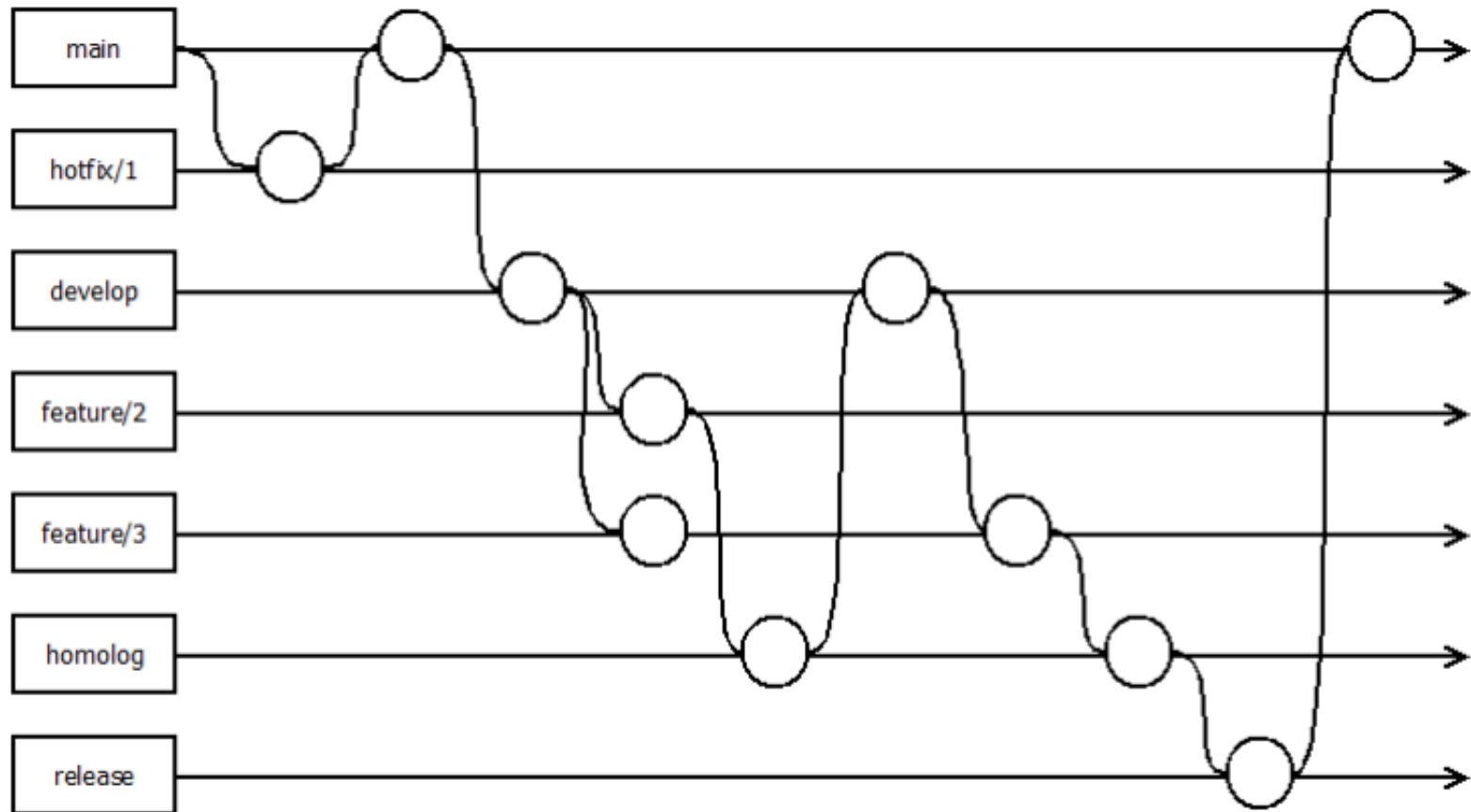
O dia a dia do desenvolvedor

- Senioridade:
 - Não graduado:
 - Estagiário.
 - Graduado:
 - Trainee;
 - Júnior;
 - Pleno;
 - Sênior;
 - Especialista.
 - Carreira em Y:
 - Supervisor/Coordenador/Gerente;
 - C-Level (CEO, COO, CFO, CIO, CTO, CMO, entre outros).

O dia a dia do desenvolvedor

- Fluxo típico de desenvolvimento:
 1. É criado um ambiente de desenvolvimento para os códigos (**develop**) a partir do ambiente principal (**main**);
 2. Os desenvolvedores codificam suas partes em seus ambientes locais (**feature/<id>**);
 3. A funcionalidade é enviada para testes na etapa de homologação (**homolog**);
 4. Caso não tenha erros, é aglutinada na etapa que será lançada no ambiente real (**release**);
 5. Tudo é mesclado no ambiente principal (**main**).
- Existe também o ambiente urgente (**hotfix/<id>**).

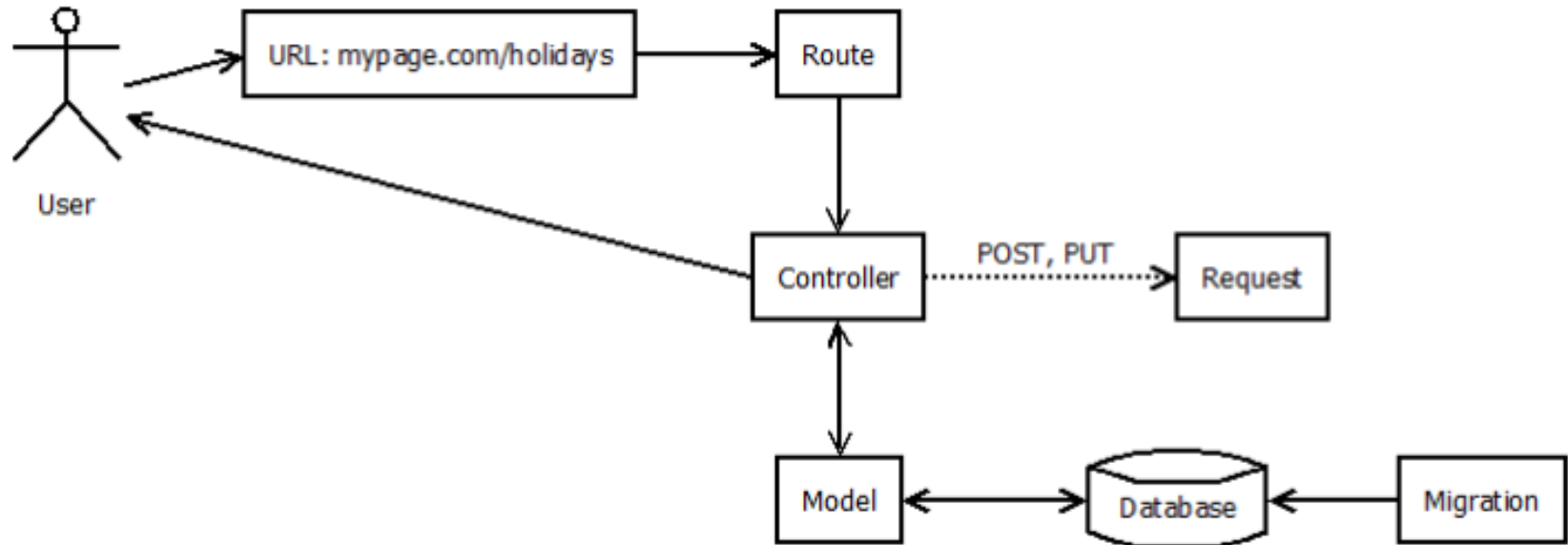
O dia a dia do desenvolvedor



Back-end com Laravel

- Framework: reúne diversas funcionalidades;
- Utiliza a linguagem de programação PHP;
- Ideal para desenvolvimento de APIs;
- Laravel básico com as funcionalidades:
 - .env: configurações locais do projeto;
 - Migrations: para criar o banco de dados;
 - Routes: endereços para operações;
 - Controllers: recebem as operações e atuam;
 - Models: manipulam os dados no banco de dados;
 - Requests: validam os dados das operações;

Back-end com Laravel



Back-end com Laravel

- Padrão de rotas de uma API RESTful:

Ação	Verbo	Rota	Método	Código	Retorno
Listar	GET	/holidays	index()	200	Array com objetos
Exibir	GET	/holidays/{id}	show()	200	Objeto específico
Cadastrar	POST	/holidays	store()	201	Objeto recém criado
Atualizar	PUT	/holidays/{id}	update()	200	Objeto recém atualizado
Excluir	DELETE	/holidays/{id}	delete()	204	Nada

- Observações:
 - Se não encontrar um recurso, retornar código 404;
 - Se a informação passada na criação ou atualização for inválida, retornar código 422 e as mensagens de erro em um array.

Testes automatizados com PHPUnit

- Testes automatizados são rápidos, robustos, flexíveis, mas não substituem testes humanos;
- Os testes devem contemplar tanto o acerto quanto os erros de cada funcionalidade;
- Quanto mais atributos no modelo existirem, mais testes deverão ser feitos, para garantir a consistência dos dados;
- Os testes não devem fazer alterações no banco de dados:
 - Se acontecer, deve ser desfeito (rollback);
 - Utilizar banco de dados para testes ou mocks.

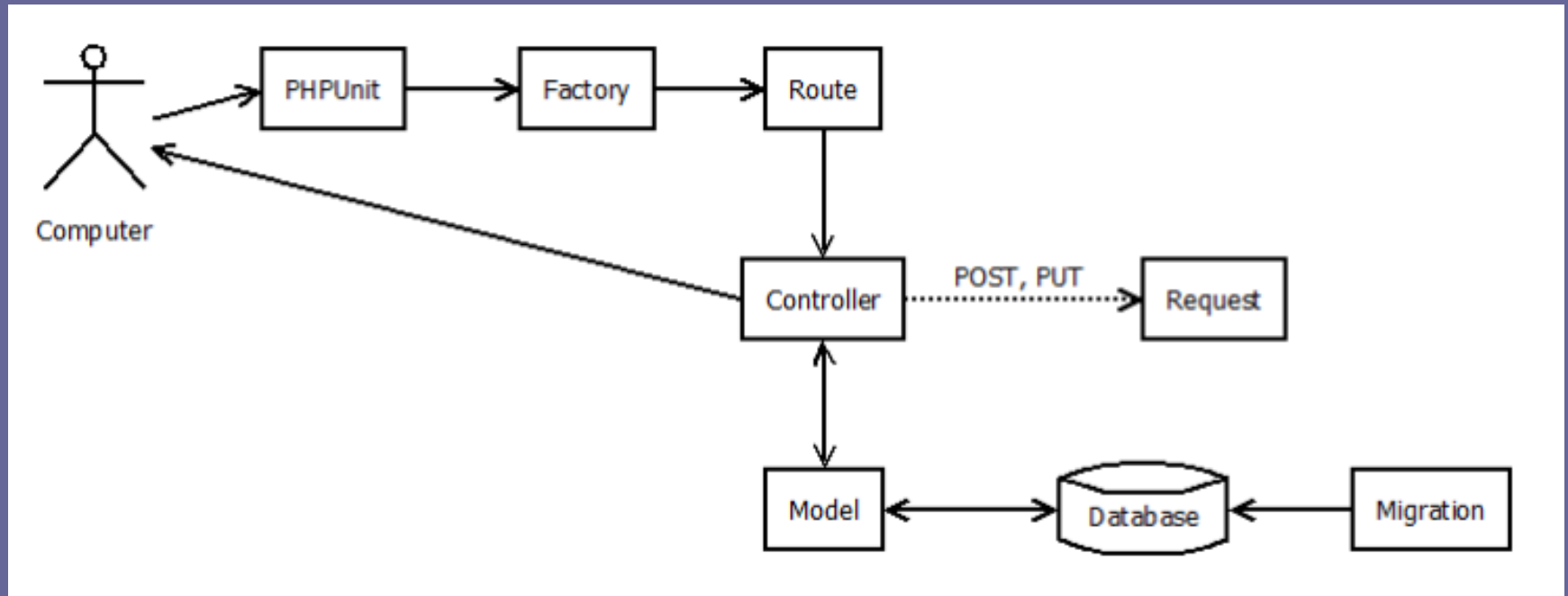
Testes automatizados com PHPUnit

- Alguns tipos de teste:
 - Unitário: a menor parte do código, geralmente um método ou função.
 - Integração: uma ou mais partes do código;
 - Sistema: todo o funcionamento do sistema;
 - Aceitação: se atende aos requisitos do usuário;
 - Regressão: garantir que o que foi feito não impacta o que já foi feito;
 - Desempenho: alto volume de dados e requisições para analisar a capacidade de carga;
 - Entre muitos outros tipos de teste.

Testes automatizados com PHPUnit

- O Laravel utiliza a ferramenta PHPUnit para os testes automatizados;
- A ferramenta PHPUnit utiliza a linguagem PHP;
- Entre tantos tipos de testes automatizados existentes, o Laravel simplificou tudo em dois:
 - **Testes Unitários**: testam pequeníssimas porções do código, como métodos de classes para um fim específico. Exemplo: um método que valida o CPF.
 - **Testes de Recursos**: testam um fluxo inteiro e um recurso desde a rota até o retorno da requisição. Exemplo: uma rota de cadastro de um produto.

Testes automatizados com PHPUnit



- **Factory**: funcionalidade para criação de dados fictícios para testes automatizados.

Cobertura de código com Xdebug

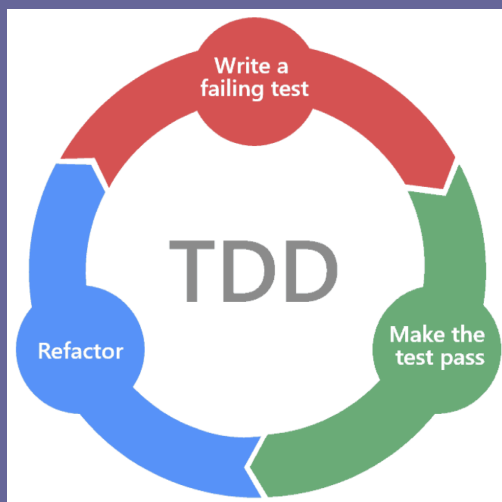
- O Xdebug é uma ferramenta poderosa para depuração de códigos em PHP;
- A cobertura de código é o percentual de linhas de código atingidas em testes automatizados;
- O ideal é atingir 100%, mas pode não ser possível, havendo redução neste limiar;
- Alcançar uma alta cobertura não significa aumentar a qualidade do código;
- Para cada if-else, será necessário um caso de teste a mais para cobrir totalmente o código.

Ao infinito e além

- Mais ferramentas:
 - PHPStan/Larastan: análise de erros e fuga de padrões e convenções do PHP (PSR's);
 - SonarQube: análise de code smells, duplicações e vulnerabilidades na segurança.
- Testes automatizados de frontend com ferramentas como Cypress, Selenium, Robot...
- Utilização em pipelines de CI/CD: é possível integrar todas estas soluções de testes e análises quando há publicação de códigos durante o projeto.

Ao infinito e além

- Test Driven Development (TDD): primeiro as classes de testes são criadas para depois as classes das funcionalidades serem criadas e passarem nos testes.
- Metodologia RGB:



- Red (vermelho): escreva o código para falhar no teste;
- Green (verde): aprimore o código para passar no teste;
- Blue (azul): refatore o código mantendo o teste sem falhar.

“Qualquer um consegue escrever
códigos que um computador
entende. Bons programadores
escrevem códigos que humanos
entendem.”

Martin Fowler