UNIVERSIDADE NOVE DE JULHO – UNINOVE

**DISCIPLINA: AUTOMAÇÃO**

IGOR NUNES ROSSETO - 409101117

LUIZ ALBERTO PEREZ JÚNIOR - 3016107194

VALDEMAR DOS REIS OLIVEIRA - 419120208

GUILHERME RAMOS - 916111476

**DEVOPS E SUA RELAÇÃO COM DESENVOLVIMENTO DE JOGOS**



São Paulo

2019

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc24459746)

[2 O QUE É DEVOPS? 4](#_Toc24459747)

[3 INTEGRANDO DEVOPS 5](#_Toc24459748)

[4 CONCEITOS - INTEGRAÇÃO CONTINUA 7](#_Toc24459749)

[5.1 FERRAMENTAS – DJANGO 8](#_Toc24459750)

[5.2 FERRAMENTAS – JENKINS 9](#_Toc24459751)

[6 INTRODUZINDO LEAGUE OF LEGENDS 10](#_Toc24459752)

[6.1 AUTOMATIZAÇÃO DE TESTES PARA LEAGUE OF LEGENDS 11](#_Toc24459753)

[7 REFERÊNCIAS BIBLIOGRÁFICAS 23](#_Toc24459754)

# 1 INTRODUÇÃO

Durante anos, 'equipes de desenvolvimento' e 'equipes de operações' foram separadas nas empresas, o que significa que as equipes de desenvolvimento se concentrariam na criação de produtos, enquanto as equipes de operação garantiriam que o código escrito pelas equipes de desenvolvimento continuasse em execução. No entanto, isso significava que levaria anos até que uma empresa pudesse lançar a próxima atualização no mercado.

No entanto, com o DevOps, as empresas de software estão vendo uma mudança drástica na maneira como não apenas desenvolvem seus produtos, mas também na maneira como testam e implantam. Segundo a Research and Markets, o DevOps deverá atingir US $ 7,72 bilhões até 2021, um crescimento de 33% ao ano. Vamos ver o que realmente é o DevOps.

# 2 O QUE É DEVOPS?

O termo "DevOps" está se tornando cada vez mais comum na indústria de software de hoje e, mesmo agora, ainda não está totalmente compreendido ou definido adequadamente. Para muitos, DevOps apenas significa a combinação das equipes de Desenvolvimento e Operação, enquanto para outros, significa automação no desenvolvimento e implantação. No entanto, embora ambas as definições estejam corretas, ambas também são igualmente limitadas.

O DevOps é um conjunto de ideologias culturais que se concentra em quebrar os silos mantidos nas empresas de software tradicionais, onde as equipes de desenvolvimento e operações eram duas equipes separadas. Ele se concentra em otimizar os processos entre as duas equipes para garantir que eles possam criar, testar e liberar software de maneira mais rápida e confiável.

O DevOps reúne entrega ágil e contínua, automação e muito mais para garantir que as equipes de software sejam eficientes, inovadoras e ofereçam maior valor a empresas e clientes. Embora muitas empresas acreditem que integrar o DevOps possa ser tão simples quanto combinar as duas equipes e colocar a automação no mix, mas, a menos que integrem o DevOps com eficiência, isso simplesmente tende a ser uma enorme bagunça!

# 3 INTEGRANDO DEVOPS

Atualmente, integrar o DevOps não é mais uma questão de opção, mas sim tempo. As empresas não têm mais a opção de não integrar o Agile ou o DevOps se quiserem que seus negócios sobrevivam. Não é uma surpresa que, com um mundo tão acelerado, as empresas agora estejam recorrendo aos especialistas em DevOps para ajudá-las a integrar o DevOps em seus negócios também.

**Existem três pilares importantes que devem ser considerados ao integrar o DevOps para o futuro do desenvolvimento de jogos:**

**Equilíbrio entre desenvolvimento e operações** - O principal objetivo do DevOps é combinar as duas equipes para garantir que elas sejam mais fortes juntas, mas se você integrar as duas equipes com idéias predestinadas, certamente encontrará um problema antes mesmo de a empresa decolar. É importante reunir as pessoas certas desde o início para garantir que elas entendam a outra equipe e realmente trabalhem juntas para criar um processo mais simplificado, com foco no produto e nos clientes.

**Ferramentas e automação** - Como o foco principal do DevOps é a entrega eficiente de um produto de qualidade, você precisa de um sistema de feedback contínuo que se concentre na movimentação de informações para frente e para trás, durante as quais o registro de log se torna uma etapa crucial. É aí que várias ferramentas e softwares de automação podem se tornar úteis. Verifique se você selecionou o software certo que torna a colaboração entre desenvolvimento, teste e implantação mais eficiente e também ajuda a economizar uma quantidade considerável de tempo.

**Crie uma mentalidade orientada à qualidade** - O principal objetivo do DevOps é criar uma sinergia entre entrega mais rápida e produto de qualidade. O Ciclo de Vida de Desenvolvimento de Software (SDLC), o DevOps requer desenvolvimento rápido de software, testes contínuos, implantação contínua e monitoramento contínuo. É aqui que você pode descartar o teste manual e, em vez disso, optar pela automação de teste para eliminar erros na fase de desenvolvimento, em vez de esperar que eles sejam empurrados para a fase de teste. No DevOps, é importante enfatizar a qualidade, em vez de simplesmente enviar o software com bugs.

# 4 CONCEITOS - INTEGRAÇÃO CONTINUA

A integração contínua é uma prática de desenvolvimento de software de DevOps em que os desenvolvedores, com frequência, juntam suas alterações de código em um repositório central. Depois disso, criações e testes são executados. Geralmente, a integração contínua se refere ao estágio de criação ou integração do processo de lançamento de software, além de originar um componente de automação (ex.: uma CI ou serviço de criação) e um componente cultural (ex.: aprender a integrar com frequência). Os principais objetivos da integração contínua são encontrar e investigar bugs mais rapidamente, melhorar a qualidade do software e reduzir o tempo que leva para validar e lançar novas atualizações de software.

# 5.1 FERRAMENTAS – DJANGO

Django é um framework para aplicações web gratuito e de código aberto, escrito em Python. Uma web framework é um conjunto de componentes que ajuda você a desenvolver sites de forma mais rápida e fácil.

Quando você está construindo um site, sempre precisa de um conjunto similar de componentes: uma maneira de lidar com a autenticação do usuário (inscrever-se, fazer login, fazer logout), um painel de gerenciamento para o seu site, formulários, uma forma de upload de arquivos, etc.

Felizmente, outras pessoas perceberam muito tempo atrás que desenvolvedores web enfrentam problemas similares quando estão construindo um novo site, então elas se juntaram e criaram frameworks (Django é um deles) que te dão componentes já prontos para usar.

Frameworks existem para evitar que você tenha que reinventar a roda e para ajudar a aliviar parte do trabalho extra quando você está construindo um novo site.

**O Django Rest Framework (DRF)** é uma biblioteca para o Framework Django que disponibiliza funcionalidades para implementar APIs Rest de forma extremamente rápida. A produtividade é o foco aqui, e acreditem, esse propósito é alcançado com louvor. Se você quiser criar uma API Rest de maneira rápida, é uma ótima ferramenta.

# 5.2 FERRAMENTAS – JENKINS

Jenkins é um servidor de automação gratuito e de código aberto. Jenkins ajuda a automatizar a parte não humana do processo de desenvolvimento de software, com integração contínua e facilitando aspectos técnicos da entrega contínua. É um sistema baseado em servidor que é executado em contêineres de servlet, como o Apache Tomcat. Ele suporta ferramentas de controle de versão, incluindo AccuRev, CVS, Subversion, Git, Mercurial, Perforce, TD / OMS, ClearCase e RTC, e pode executar projetos baseados no Apache Ant, Apache Maven e sbt, bem como scripts de shell arbitrários e comandos em lote do Windows. O criador de Jenkins é Kohsuke Kawaguchi. Lançado sob a licença MIT, Jenkins é um software livre.

As builds podem ser acionadas por vários meios, por exemplo, commit em um sistema de controle de versão, agendar através de um mecanismo semelhante ao cron e solicitar uma URL de uma build específica. Também pode ser acionado após a conclusão de outras builds na fila. A funcionalidade do Jenkins pode ser estendida com plugins.

# 6 INTRODUZINDO LEAGUE OF LEGENDS

League of Legends (abreviado como LoL) é um jogo eletrônico do gênero multiplayer online battle arena, desenvolvido e publicado pela Riot Games para Microsoft Windows e Mac OS X. É um jogo gratuito para jogar e inspirado no modo Defense of the Ancients de Warcraft III: The Frozen Throne.

Em League of Legends, os jogadores assumem o papel de "invocadores", controlando campeões com habilidades únicas e que lutam com seu time contra outros invocadores ou campeões controlados pelo computador. Os campeões utilizam como comandos os botões do teclado Q, W, E, R, D, F e de 1 a 6 para executar habilidades. No modo mais popular do jogo, o objetivo de cada time é destruir o nexus da equipe adversária, uma construção localizada na base e protegida por outras estruturas. Cada partida de League of Legends é distinta, pois os campeões sempre começam fracos e progridem através da acumulação de ouro e da experiência ao longo do jogo.

# 6.1 AUTOMATIZAÇÃO DE TESTES PARA LEAGUE OF LEGENDS

**Aplicando DevOps**

League muda muito, muito rapidamente. Em média, ocorrem mais de 100 alterações de código e conteúdo verificadas no controle de origem todos os dias, e fornecer cobertura adequada para todas essas alterações é um desafio. Com um novo patch a cada duas semanas, é fundamental que sejam detectados defeitos rapidamente. Os erros descobertos no final do processo de lançamento podem causar atrasos, levar a reimplementações ou exigir desabilitações temporárias dos campeões - todas as experiências ruins para os jogadores. A automação libera os analistas de qualidade para se concentrarem em testes mais criativos e na prevenção de defeitos.

A automação também fornece retorno mais rápido para os resultados dos testes. Não é viável para humanos executar uma varredura completa de teste em cada novo código ou envio de conteúdo e, mesmo que fosse, seria necessário que um exército de testadores retornasse os resultados rapidamente.

O sistema de teste é executado em integração contínua (CI) e os relatórios são retornados cerca de uma hora após o check-in. Isso significa que os desenvolvedores recebem resultados em um prazo razoável, o que ajuda a reduzir a alternância de contexto; de fato, os bugs descobertos na automação são resolvidos oito vezes mais rápido que os bugs comuns. Melhor ainda, se for preciso aumentar a taxa de transferência para testes, pode-se simplesmente adicionar mais alguns executores ao farm de testes.

**O sistema de verificação de build**

O Sistema de Verificação de build (BVS), é a estrutura de teste para o cliente e servidor do jogo. É responsável por adquirir artefatos para testar, implantá-los em uma máquina de teste, iniciar e gerenciar sistemas em teste, executar testes e relatar seus resultados. Os testes são escritos em Python, e a maior parte do código do é BVS para isolar os gravadores de teste das complexidades de reunir os recursos necessários. Como resultado, alguns argumentos em uma classe de teste podem especificar qual mapa executar, quantos clientes incluir e quais campeões devem estar no jogo.

Os testes usam remote procedure call (RPC) endpoints expostos no cliente e no servidor para emitir comandos e monitorar o estado do jogo. Na maioria das vezes, os testes consistem em um conjunto bastante linear de instruções e consultas - os testes existentes abrangem tudo, desde as habilidades dos campeões às regras de visão até as recompensas esperadas pela morte de um lacaio. Alguns dos testes anteriores eram significativamente menos lineares, mas isso tornava o trabalho com o sistema muito mais difícil para desenvolvedores menos técnicos.

Como todo o trabalho de configuração de uma área de trabalho de teste é realizado separadamente, os próprios testes devem ser iguais, seja executando em uma área de trabalho local ou em no farm de testes. Isso facilita a execução de testes localmente enquanto se faz alterações no jogo.

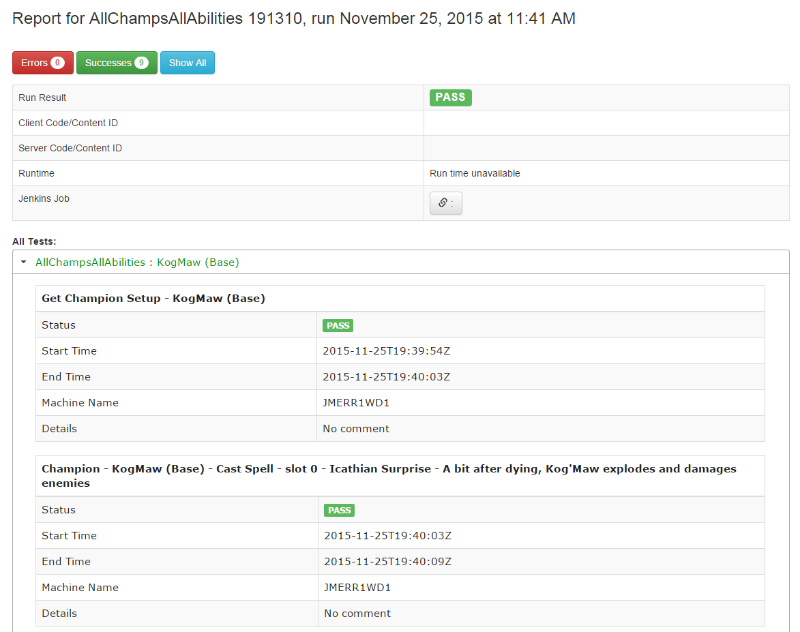
A seguir, um exemplo do teste para o dano causado pelo novo W de Kog'Maw (Campeão do jogo) é assim:

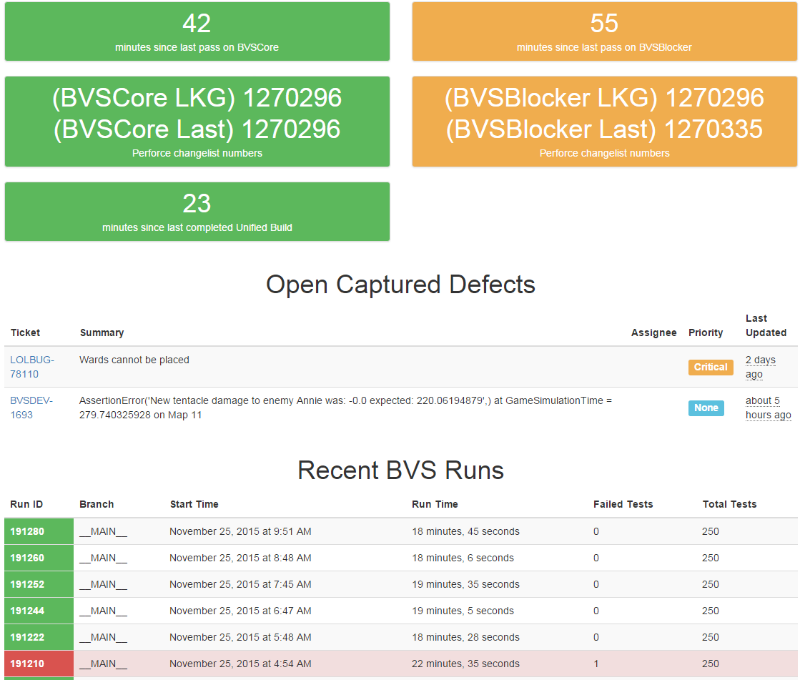
****

A primeira parte do conjunto de testes de Kog’Maw, incluindo o teste de dano da Barragem Arcana, é a seguinte(vídeo):



Quando um teste termina uma execução, ele fornece os resultados para um serviço de relatório separado, que armazena dados de execução que remontam aproximadamente seis meses. Dependendo da fonte dos dados de teste fornecidos, este serviço executa ações diferentes. Uma execução local de um teste abre uma página da Web na máquina em execução que detalha os casos de aprovação e reprovação. Uma execução no farm de teste, no entanto, criará novos tickets de bug para quaisquer problemas descobertos, marcará os artefatos de acordo com os resultados e enviará um email aos confirmadores, se houver casos com falha. Os dados de teste também são agregados e rastreados por meio do serviço de relatório, permitindo ver quando ocorreram falhas no teste, com que frequência elas ocorrem e quanto tempo se passou desde a última build.



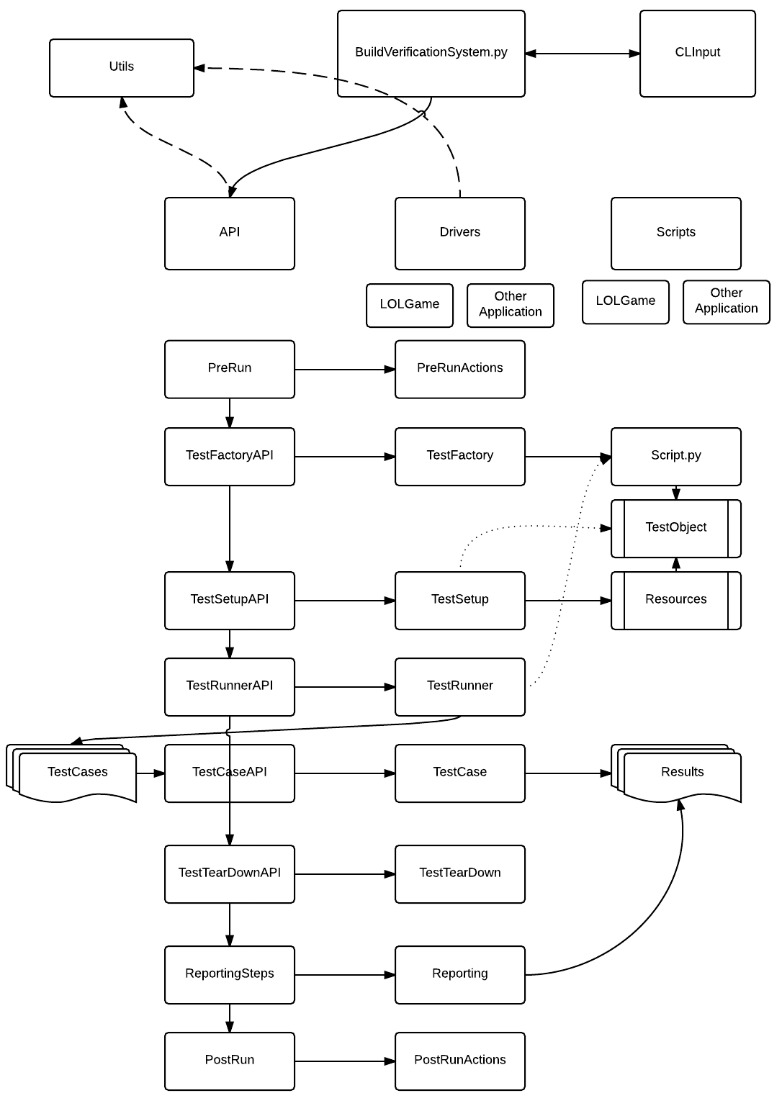


Para evitar testes inadequados ou não confiáveis, cada um deve passar por um processo padrão para ser confiável. Depois que um teste é revisado e enviado ao código, ele entra em um conjunto de testes chamado BVSStaging. Lá, os testes devem demonstrar estabilidade por pelo menos uma semana antes de serem promovidos. Se um teste na preparação falhar, apenas os desenvolvedores do teste serão notificados para evitar confusão.

Depois que um teste demonstra sua confiabilidade, ele é promovido em um dos dois conjuntos. O primeiro conjunto, BVSBlocker, contém testes que indicam se uma construção vale a pena testar ainda mais. Uma compilação que falha no Blocker não é implantada em um ambiente de teste, porque os jogos não estão sendo iniciados ou existem vários bugs graves que afetam o jogo. Sua contraparte, BVSCore, é o conjunto principal de testes funcionais, incluindo testes para todas as habilidades de campeão.

**Conhecendo o Framework**

**O BVS é implementado em três camadas: o executor, os drivers e os scripts. O executor implementa uma API genérica para teste funcional, enquanto os drivers implementam as etapas específicas de configuração e execução de um teste. Por fim, os scripts implementam a lógica específica para casos de teste. Atualmente, há apenas um driver em uso (LOLGame), mas a separação executor-driver significa que projetos futuros podem usar o BVS implementando seu próprio driver e usando utilitários compartilhados escritos quando o driver LOLGame foi escrito.**

****

Componentes individuais registram seus argumentos obrigatórios e opcionais como parte de sua declaração. Quando os argumentos são fornecidos na linha de comando, eles são armazenados como um dicionário que os componentes consomem como parte de sua inicialização. As versões anteriores do BVS usavam a biblioteca argparse padrão para Python, mas foi optado por se afastar do argparse por dois motivos: primeiro, o número de entradas potenciais de argumentos estava se tornando grande e, portanto, muito difícil de rastrear através do sistema; e, segundo, os drivers precisavam ter argumentos específicos para o driver, o que significava que declarar um parser na inicialização não era viável.

****

Exemplo de argumentos para um objeto driver

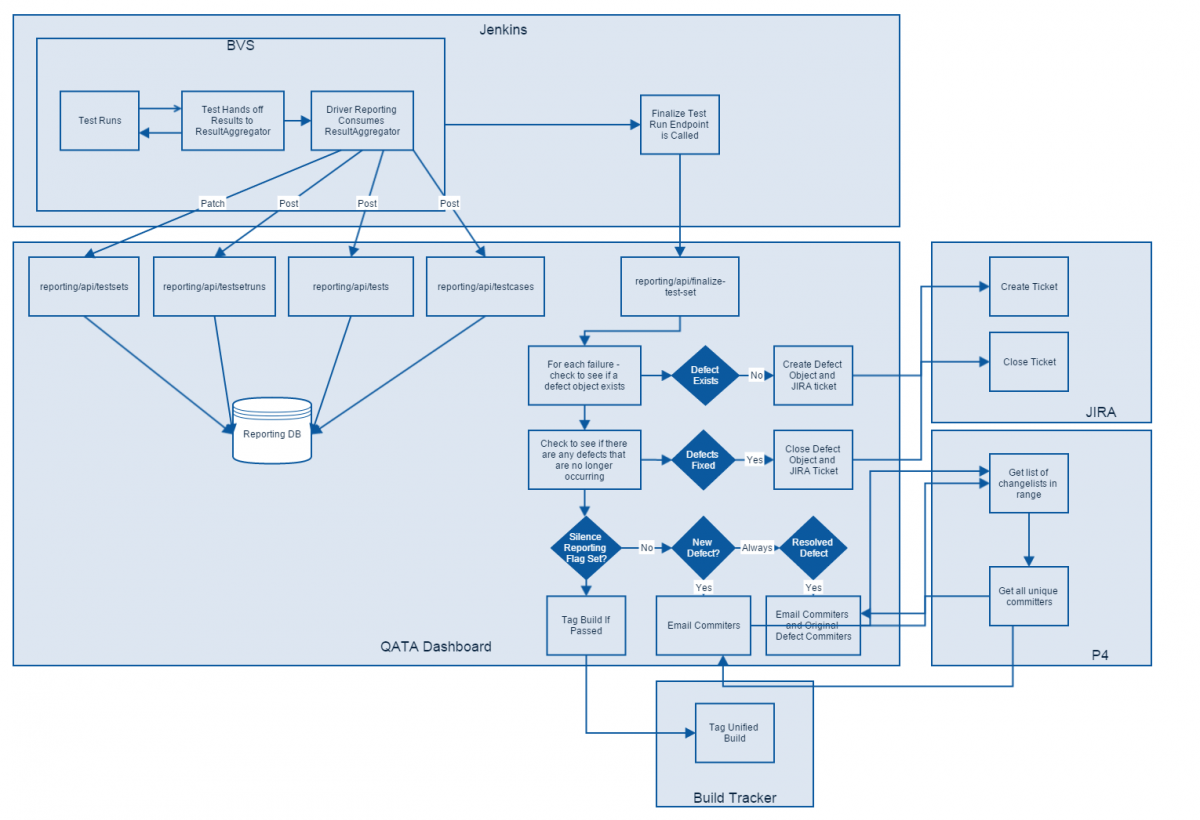
Existem três níveis de granularidade relevantes: conjuntos de testes, testes e casos de teste.

**Conjuntos de testes** são um grupo de testes que são executados juntos. Por exemplo, o conjunto de testes BVSBlocker mencionado anteriormente é um conjunto de testes de fumaça que são executados em CI. Os conjuntos de testes são atualmente descritos no BVS por meio de arquivos JSON que podem ser criados no VCS ou em movimento.

**Testes** são classes individuais que implementam um conjunto de casos de teste semelhantes que usam a mesma configuração básica de jogo. Por exemplo, o teste LoadChampsAndSkins é executado em casos de teste que consistem em carregar os ativos para cada campeão e skin e confirmar que a carga ocorre corretamente.

**Os casos de teste** são unidades únicas da funcionalidade esperada dentro de um teste. Por exemplo, a função loadChampionAndSkin no teste LoadChampsAndSkins é um único caso de teste que é executado centenas de vezes para cobrir cada combinação de campeão e skin. Todo o caso de teste do Kog'Maw acima é executado por um teste de nível superior, que permite que casos de teste mais complicados tenham um pouco mais de estrutura do que uma função.

Uma das outras grandes adaptações que foram feitas no BVS desde os primeiros dias é separar a lógica de tudo, exceto executar um teste. No passado, o BVS fazia o trabalho de descobrir quais artefatos ele deveria usar, marcar as construções como aprovadas ou reprovadas e compor seus próprios relatórios de teste. Para preservar uma clara separação de responsabilidades, há um serviço separado que faz todo o trabalho que não pertence à execução direta de testes. O serviço é um aplicativo Django usando o Django REST Framework para fornecer uma API que o BVS e outros serviços atingem no estado atual do BVS.



Fluxograma de teste e pós-teste

<https://technology.riotgames.com/sites/default/files/bvs_flowchart2_0.png> - Ampliado

**Concluindo - Desempenho geral**

No geral, o BVS executa ~ 5500 casos de teste em aproximadamente 18 minutos para cada nova versão do League of Legends. No total, são cerca de 100.000 casos de teste por dia. O tempo médio entre o envio do defeito e o primeiro relatório de falha do BVS é de uma a duas horas. 50% de todos os erros críticos ou no nível do bloqueador são descobertos pelo BVS, sendo o restante descoberto durante o controle de qualidade interno ou no PBE (Public Beta Enviroment). Os problemas não detectados pelo BVS geralmente desaparecem devido à falta de cobertura e não a testes ruins.

Enquanto a maioria dos bugs descobertos caem na parte de crashes ou falta de funcionalidades, ocasionalmente são encontrados alguns bugs realmente excelentes. Um deles era o defeito em que todas as torres do jogo deslizavam lentamente para o canto superior direito do mapa, resultando em um épico engarrafamento de torres na base do lado roxo. Outras descobertas também incluem coisas que os testes não automatizados não teriam necessariamente descoberto, como um problema em que os tiros de habilidade passariam por um inimigo se um campeão atingisse esse inimigo exatamente à queima-roupa.

Como um todo, o teste automatizado não substituiu necessariamente o manual, mas ajudou a acelerar o ciclo de feedback de desenvolvimento e liberou mais testadores manuais para se concentrar em testes destrutivos. À medida que mais conteúdo é adicionado ao League of Legends, mais cobertura é adicionada, o que deve aumentar a taxa de acertos por defeitos e melhorar a confiança na saúde das builds.

# 7 REFERÊNCIAS BIBLIOGRÁFICAS

DevOps -

https://blog.eduonix.com/software-development/game-development-future-devops/

Continuous Integration

https://aws.amazon.com/pt/devops/continuous-integration/

Django e Django REST Framework

https://tutorial.djangogirls.org/pt/django/

Jenkins

https://en.wikipedia.org/wiki/Jenkins\_(software)

Artigo em Riot Games:

Automated Testing for League of Legends

https://pt.wikipedia.org/wiki/League\_of\_Legends

https://technology.riotgames.com/news/automated-testing-league-legends