1. **Descrição do App e da Empresa**

Fundado em 2003, o KaBuM! foi um dos pioneiros no comércio eletrônico brasileiro e hoje é o maior e-commerce de Tecnologia e Games na América Latina. Desde sua criação, a empresa é formada por histórias e conquistas de um time obcecado por agilidade, qualidade de atendimento, velocidade de entrega e respeito pelo consumidor.

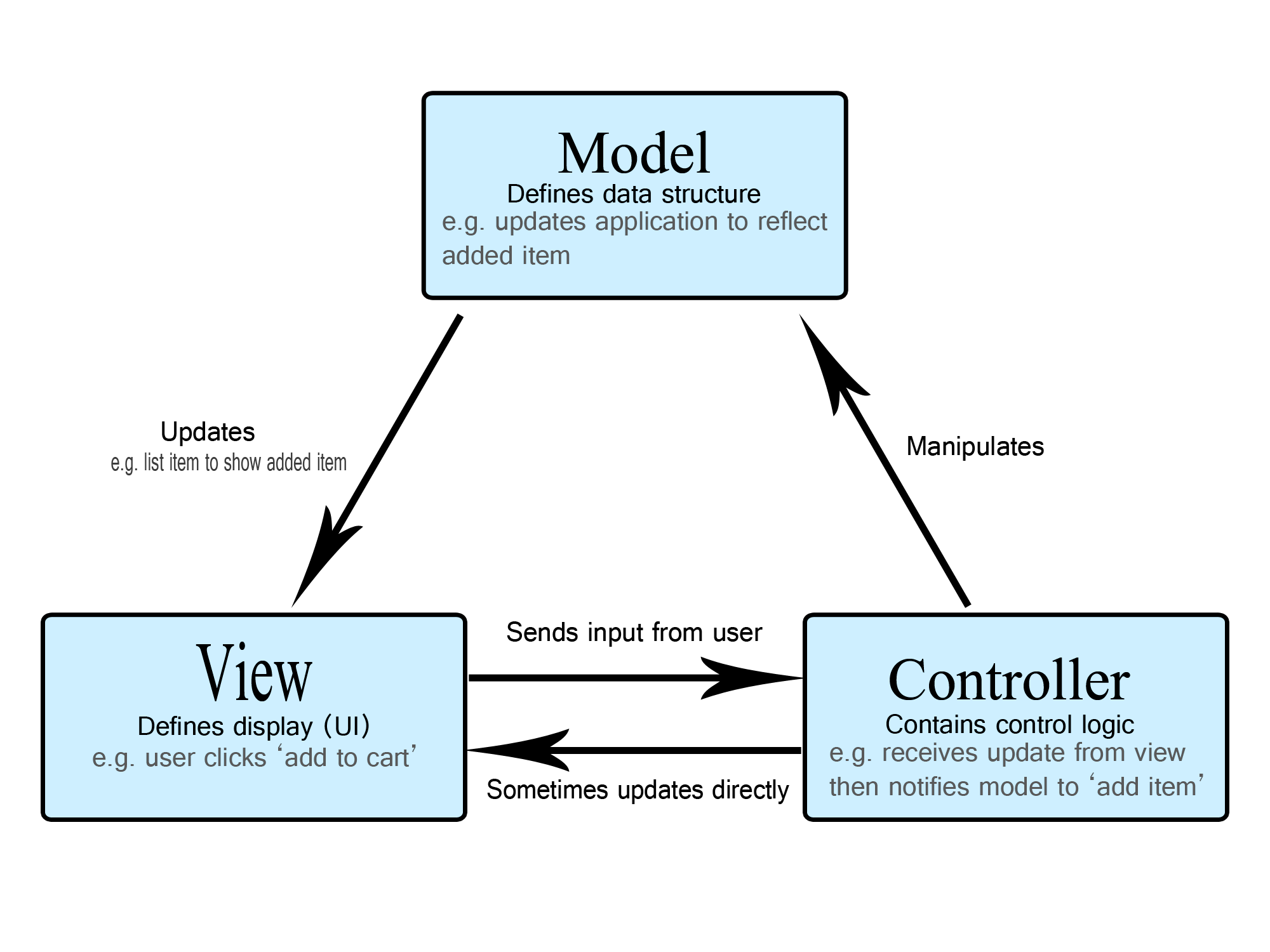
O foco da companhia é o comércio digital e oferece mais de 20 mil itens de tecnologia de ponta para profissionais e para o universo gamer. A plataforma superou 8 milhões de clientes e faz entregas em cerca de 5 mil cidades. Seu público alvo são gamers fãs de tecnologia.

A atuação da companhia se estende ao incentivo de esportes eletrônicos. Entre os games já patrocinados, estão: Counter-Strike e FIFA e Free Fire e também é responsável pela KaBuM! Esports, que é uma das principais participantes do campeonato Brasileiro League of Legends (CBLoL). É tetracampeã nacional e a primeira representante do país no campeonato mundial.

A empresa já recebeu várias certificações e se destacou em ranking significativos. Em 2018, teve o segundo maior índice de recomendações pelos consumidores brasileiros e em 2021 conquistou o primeiro lugar no prêmio Reclame Aqui e se tornou uma empresa GPTW (Great Place to Work).

1. **Arquitetura do projeto**

MVC (Model-View-Controller)



1. **Padrão para nomenclatura de código**

**Nomes de classes:** Os nomes de classes têm a primeira letra de cada palavra maiúscula (CamelCase).

Ex: NomeDeUmaClasse

**Nomes de funções:** Nomes de funções e métodos devem estar em letras minúsculas, com palavras separadas por *underscores* conforme seja útil para a legibilidade.

Ex: nome\_de\_uma\_funcao

**Constantes:** Constantes são geralmente definidas em um nível de módulo e escritas em letras maiúsculas com *underscores* separando as palavras.

Exemplos: MÓDULO ESTATÍSTICO

**Padrões e nomenclaturas no Git**

**bugfix**: Como o próprio nome já diz, é um BUG e precisa ser corrigido de forma imediata, o quanto antes. Num outro artigo eu explico melhor a utilização desse cara e *branches principais*.

Ex: bugfix: commit

**feature:** O nome já diz também o que é, uma nova feature que será adicionada ao projeto, componente e afins.

Ex: feature: commit

**hotfix:**Ele é bem similar ao bugfix/, porém, ele não é um BUG, mas sim uma correção, seja ela de cor, textos, alterações não tão urgentes, que não signifiquem BUG's.

Ex: hotfix: commit

**Resumo dos Testes:**

* Acessar o site da KaBuM! como visitante; (Acessar o site com a tela Maximizada e Acessar com tela reduzida)
* Acessar o Menu Oferta do dia; (Pelo Menu ou Pelo campo de busca)
* Acessar o Menu Departamento > Computadores;
* Adicionar um monitor no carrinho de compras.

**Caso de Testes:**

CT1 - Acessar o site com a tela maximizada >

CT2 - Acessar com tela reduzida >

CT3 - Acessar Oferta do Dia pelo Menu >

CT4 - Acessar Oferta do Dia pelo campo de busca >

CT5 - Acessar Departamento > Computadores

CT6 - Adicionar um monitor no carrinho de compras

Obs: Ao acessar o menu DEPARTAMENTOS com dispositivo móvel ou com tela reduzida, a plataforma redireciona para o departamento GAMES.

**Ferramentas para Automação de Testes:**

* Selenium;
* Cypress;
* Jenkins.

**Estratégia de deploy:**

* **Rolling Upgrade**

A estratégia escolhida foi a Rolling Upgrade, tendo em vista que ela permite a atualização gradual do software e a detecção rápida de problemas, minimizando o risco de falhas em produção.

**Importância de code review:**

O code review é uma etapa essencial para o desenvolvimento de softwares, pois permite que o time analise as alterações que serão submetidas no repositório do projeto e garantir que as mesmas estejam de acordo. Outra característica do code review é que, para fazer os ajustes, geralmente, é chamado um outro desenvolvedor, diferente do que está criando os códigos, dessa forma obtendo uma segunda opinião sobre o que está sendo executado e dominuindo a possibilidade de erros.

**Montem uma timeline contendo os pontos:**

**Planejamento:**

* Definir os requisitos e objetivos dos testes.
* Identificar os cenários de teste e critérios de aceitação.
* Estabelecer os recursos e prazos necessários.

**Desenvolvimento**

* Criar os casos de teste com base nos cenários definidos.
* Desenvolver os scripts de teste ou preparar o ambiente de teste manual.
* Verificar se todas as dependências necessárias para os testes estão disponíveis.

**Geração de build**

* Compilar o código-fonte do software em uma versão executável.
* Realizar a integração e teste unitário para verificar a estabilidade e funcionalidade básica.

**Deploy em ambiente de homologação**

* Instalar a build gerada em um ambiente de homologação separado.
* Executar os testes de integração e teste de sistema.
* Realizar testes de desempenho, segurança e usabilidade, se aplicável.
* Registrar e relatar os resultados dos testes.

**Deploy em produção**

* Preparar o ambiente de produção para receber a nova versão.
* Executar testes finais de aceitação em um ambiente espelhado ao de produção.
* Verificar a compatibilidade com outros sistemas em produção.
* Realizar o deploy da nova versão para o ambiente de produção.
* Monitorar e validar o funcionamento do sistema em produção.

**Onde o software poderia ser testado e de que forma?**

De acordo com a timeline apresentada, o software poderia ser testado em dois ambientes: homologação e produção.

No ambiente de homologação, o software poderia ser testado através de testes automatizados, como testes de integração, testes de aceitação do usuário, testes de desempenho, entre outros, usando ferramentas como o Selenium, Cypress e outras.

Já no ambiente de produção, o software poderia ser testado através de monitoramento contínuo, que permite identificar problemas em tempo real, usando ferramentas de monitoramento como Nagios, Zabbix e outras.

**Usando a mesma timeline, estratégia CI/CD**

**Planejamento:**

* Utilizar ferramentas de gerenciamento de projetos para definir e acompanhar os requisitos e objetivos dos testes.
* Manter um repositório de documentação atualizado para compartilhar os cenários de teste e critérios de aceitação.

**Desenvolvimento**

* Utilizar um sistema de controle de versão, como o Git, para gerenciar o código-fonte e as alterações.
* Utilizar ferramentas de automação de teste, como frameworks de teste unitário, para desenvolver e executar os casos de teste automaticamente.
* Integrar essas ferramentas de teste com o processo de construção automatizado.

**Geração de build**

* Utilizar ferramentas de automação de build, como o Jenkins ou o CircleCI, para automatizar a geração de builds após a conclusão das tarefas de desenvolvimento e revisão de código.
* Utilizar técnicas de empacotamento, como o Docker, para gerar imagens do software que possam ser facilmente distribuídas.

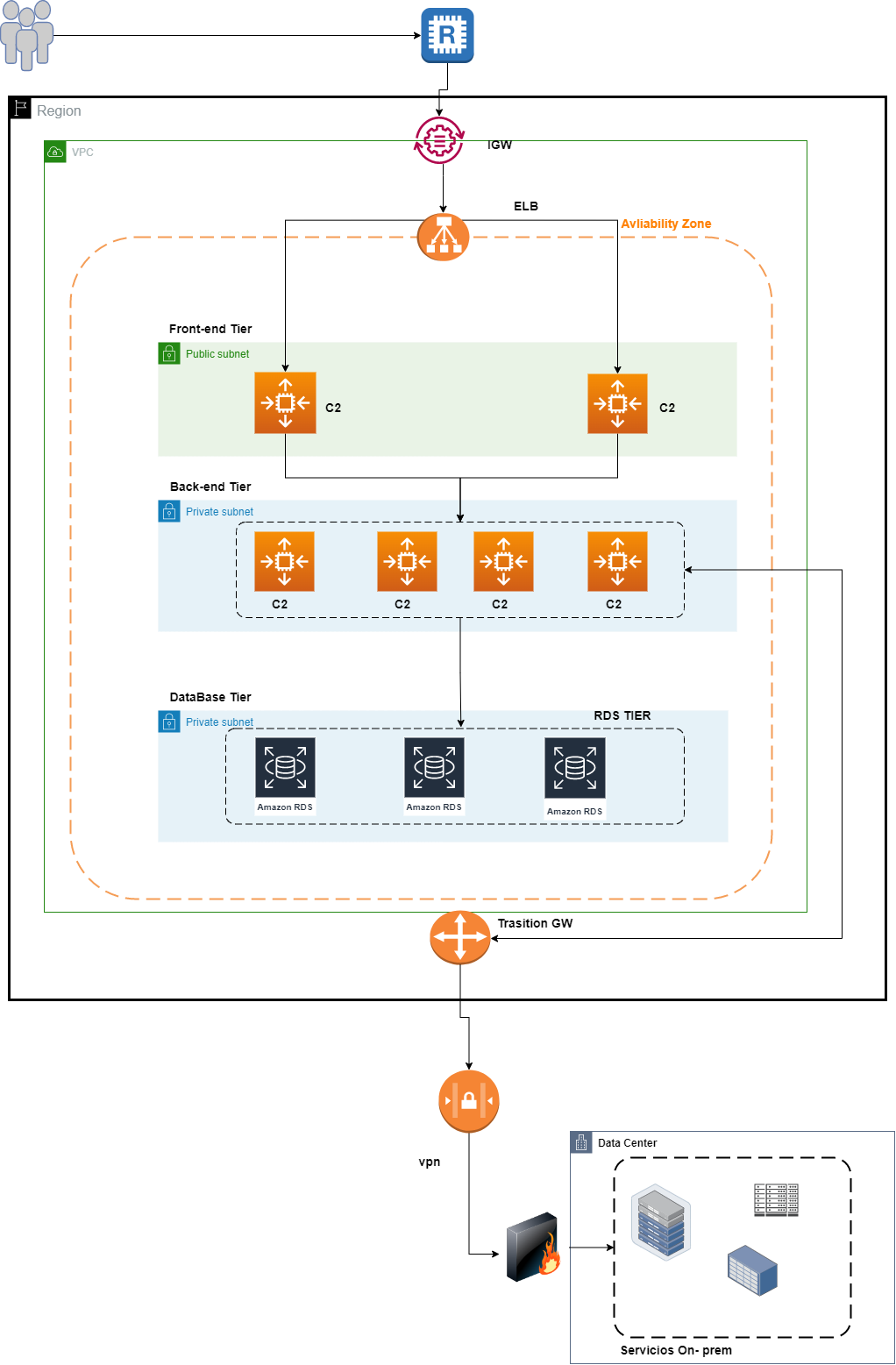
**Deploy em ambiente de homologação**

* Utilizar ferramentas de automação de deploy, como o Ansible ou o Chef, para realizar o deploy do software em um ambiente de homologação.
* Utilizar ferramentas de automação de testes, como o Selenium ou o Cypress, para automatizar a execução dos casos de teste definidos anteriormente.

**Deploy em produção**

* Utilizar ferramentas de automação de deploy, como o Ansible ou o Chef, para realizar o deploy do software em ambiente de produção.
* Utilizar ferramentas de monitoramento, como o Nagios ou o Zabbix, para monitorar o sistema e identificar possíveis problemas.

**Diagrama dos componentes**



Link de acesso: <https://drive.google.com/file/d/1MabUz6zt0Of79gfSNE8lWl3uz78wBUQ9/view?usp=sharing>

**DESCRIÇÃO DOS COMPONENTES DO DIAGRAMA**

**ROUTE53**

O Route 53 é um serviço web de sistema de nome de domínio (DNS) em nuvem altamente disponível e escalável fornecido pela Amazon Web Services (AWS).

Ele fornece aos desenvolvedores e empresas um método confiável de roteamento de usuários finais para aplicativos da Internet, traduzindo nomes de domínio (como exemplo.com) em endereços IP (como 192.0.2.1), que os computadores usam para se identificar na Internet.

O Route53 oferece vários recursos como por exemplo: Registro e gerenciamento de domínio, roteamento de DNS, Verificação de integridade, gerenciamento de fluxo, entre outras funções...

No geral, o Route 53 é um serviço DNS poderoso e flexível que ajuda a melhorar a disponibilidade, a confiabilidade e o desempenho dos aplicativos da Internet.

**CloudFront**

O AWS CloudFront é um serviço de distribuição de conteúdo (CDN - Content Delivery Network) altamente escalável e confiável que acelera a entrega de conteúdo aos usuários finais, melhorando o desempenho, a segurança e a experiência do usuário em aplicações e sites da web.

O CloudFront utiliza uma rede global de servidores em vários locais estratégicos, conhecidos como edge locations, para armazenar em cache e distribuir o conteúdo aos usuários com baixa latência. Quando um usuário solicita um recurso, o CloudFront roteia a solicitação para a edge location mais próxima, reduzindo o tempo de resposta e melhorando a experiência do usuário.

Principais características:

- Alta velocidade de entrega;

- Escalabilidade;

- Segurança;

- Flexibilidade;

- Integração com serviços da AWS;

- Monitoramento e análise.

**ELB**

A AWS ELB (Elastic Load Balancer) é um serviço que distribui de forma equilibrada o tráfego de rede recebido entre vários destinos, como instâncias EC2, contêineres ou endereços IP. Ele melhora a disponibilidade, tolerância a falhas e escalabilidade de aplicativos, atuando como um único ponto de entrada para clientes e roteando automaticamente o tráfego para instâncias saudáveis. A ELB oferece diferentes tipos de balanceadores de carga, incluindo o Classic Load Balancer (CLB), Application Load Balancer (ALB) e Network Load Balancer (NLB), cada um atendendo a requisitos específicos. A ELB fornece recursos como alta disponibilidade, escalabilidade, terminação SSL, verificação de integridade, persistência de sessão e capacidades de monitoramento. Ela desempenha um papel fundamental na construção de arquiteturas resilientes e de alto desempenho na nuvem da AWS.

**EC2**

O AWS EC2 (Elastic Compute Cloud) é um serviço de computação em nuvem que permite que você alugue servidores virtuais, chamados de instâncias, para executar suas aplicações na nuvem. O EC2 oferece escalabilidade, flexibilidade e recursos poderosos para criar e gerenciar facilmente ambientes de computação na nuvem. Com o EC2, você pode provisionar, configurar e controlar instâncias de acordo com suas necessidades de computação, pagando apenas pelo tempo de utilização. Ele também oferece integração com outros serviços da AWS e fornece recursos avançados, como balanceamento de carga, auto scaling e armazenamento persistente.

**RDS**

Amazon RDS (Relational Database Service) é um serviço gerenciado que torna fácil configurar, operar e dimensionar um banco de dados relacional na nuvem. As réplicas de leitura RDS e os bancos de dados primários RDS são dois componentes diferentes do Amazon RDS.

**RDS Read Replicas**

As réplicas de leitura do RDS, por outro lado, são cópias somente leitura da instância de banco de dados primária. Elas são usadas para descarregar o tráfego de leitura da instância primária e fornecer escalabilidade de leitura. As réplicas de leitura recebem atualizações de dados da instância de banco de dados primária quase em tempo real usando replicação assíncrona. Isso significa que as réplicas de leitura podem ser usadas para lidar com cargas de trabalho de leitura pesadas, liberando a instância primária para lidar com cargas de trabalho de gravação pesadas.

**RDS Primary Databases**

Os bancos de dados primários do RDS são as instâncias de banco de dados primárias em um cluster de bancos de dados que lidam com todas as operações de gravação e algumas operações de leitura. Eles são a fonte principal da verdade para os dados armazenados no banco de dados. A instância de banco de dados primária é responsável por sincronizar dados com as réplicas de leitura.

**Standby Databases**

É uma segunda Database com a cópia do sistema em produção atualizado que as empresas utilizam para proteger os seus dados, em casos de perdas na sua Database primária, seja por falha no hardware ou por erros humanos. No caso de desastres a empresa pode apontar as suas aplicações para esse backup, não interrompendo o serviço para seus clientes.

**Transit GW**

É um dispositivo intermediário entre 2 ou mais redes. Em uma rede de trânsito, o tráfego é encaminhado de uma rede para outra por meio do gateway. O gateway é responsável por rotear o tráfego com base no endereço de destino, bem como por executar várias outras funções de rede, como segurança, gerenciamento de qualidade de serviço (QoS) e conversão de endereço de rede (NAT).

No contexto da computação em nuvem, onde fornecem um meio de conectar nuvens privadas virtuais (VPCs) à internet pública e outros VPCs. Isso permite que as organizações criem arquiteturas de nuvem híbrida que combinam a escalabilidade e a flexibilidade da nuvem com a segurança e o controle de suas redes privadas.

**Cloud Watch**

É um serviço de monitoramento e observabilidade, que permite que os usuários coletem e rastreiem métricas, coletem e monitorem arquivos de log e defina alarmes. O CloudWatch pode monitorar recursos da AWS, como instâncias do Amazon EC2, instâncias de banco de dados do Amazon RDS e Elastic Load Balancers, bem como métricas personalizadas geradas por um aplicativo. Ele fornece uma visão centralizada dos recursos e aplicativos da AWS executados na AWS, bem como a capacidade de automatizar tarefas operacionais com base em condições específicas, como escalar um aplicativo em resposta à demanda. O CloudWatch também fornece insights e visualizações sobre o desempenho de todo o sistema e a utilização de recursos.

**VPN**

VPN significa Rede Privada Virtual. É uma tecnologia que permite criar uma conexão de rede segura e privada em uma rede pública, como a internet.

Uma VPN criptografa seu tráfego de internet e o encaminha por meio de um servidor remoto. Isso permite que você acesse a Internet como se estivesse em um local diferente, e suas atividades online ficam ocultas do seu provedor de serviços de Internet (ISP) e de terceiros.

As VPNs são comumente usadas para proteger dados confidenciais e privacidade online, bem como para contornar restrições geográficas e censura. Eles também são frequentemente usados por funcionários remotos para acessar com segurança os recursos da empresa e se comunicar com os colegas.

**FireWall**

Um firewall é um dispositivo de segurança de rede que monitora e controla o tráfego de rede de entrada e saída com base em regras de segurança predeterminadas. Seu objetivo principal é impedir o acesso não autorizado de ou para uma rede privada enquanto ainda permite a passagem de comunicação autorizada. Os firewalls podem ser baseados em hardware, software ou uma combinação de ambos. Eles usam várias técnicas, como filtragem de pacotes, inspeção de estado e gateway em nível de aplicativo, para analisar o tráfego de rede e determinar se devem ser permitidos ou bloqueados com base em políticas de segurança predefinidas. Os firewalls são um componente essencial de uma estratégia abrangente de segurança de rede e são comumente usados em ambientes domésticos e empresariais para proteção contra ataques cibernéticos e outras ameaças à segurança.

**Serviços On-Premise**

On-Premise ou “na premissa" são servidores locais, é um tipo de sistema de informação para ERP (Enterprise Resource Planning) em que os dados e os processos de um negócio são armazenados e gerenciados na própria organização.

**DOCUMENTO DE DISASTER RECOVERY**

O plano de disaster recovery (recuperação de desastres) é essencial para garantir a continuidade dos negócios em caso de interrupções graves, como desastres naturais, falhas de hardware, ataques cibernéticos ou outros eventos catastróficos. Aqui estão alguns cenários comuns que podem exigir a implementação de um plano de disaster recovery:

**CENÁRIO E RESOLUÇÃO 01:**

**Cenário:** Desastres naturais: Incêndios, terremotos, inundações, furacões e tempestades severas podem causar danos significativos às instalações físicas de uma organização, interrompendo as operações normais. Um plano de disaster recovery nesse cenário pode incluir a transferência das operações para um local alternativo, backup e recuperação de dados, bem como a reconstrução ou reparo das instalações afetadas.  
 **Resolução:** Desastres naturais: Ter um plano de recuperação de desastres para eventos como terremotos, inundações, incêndios ou tempestades é crucial. Nesses casos, a solução pode envolver a implementação de um site de recuperação de desastres (DR site) em uma localização geograficamente distante. Esse site deve ter infraestrutura de TI e sistemas de armazenamento de dados redundantes para que as operações comerciais possam ser retomadas rapidamente.

**CENÁRIO E RESOLUÇÃO 02:**

**Cenário:** Falhas de hardware ou software: O mau funcionamento de servidores, falhas de energia, corrupção de dados ou falhas de software podem resultar em interrupções operacionais. Um plano de disaster recovery para esse cenário pode envolver a disponibilidade de sistemas de backup em tempo real, a replicação de dados para servidores secundários ou a implementação de nuvens de recuperação.

**Resolução:** Falha de hardware ou sistema: Se ocorrer uma falha em um servidor, sistema de armazenamento ou outro componente crítico, a solução pode envolver a implementação de redundância e backups regulares. Isso pode incluir a configuração de servidores espelhados ou em cluster, onde um sistema pode assumir as funções do outro em caso de falha. Além disso, ter backups atualizados e testados regularmente é essencial para restaurar os dados e os sistemas operacionais em uma nova infraestrutura.

**CENÁRIO E RESOLUÇÃO 03:**

**Cenário:** Ataques cibernéticos: Incidentes de segurança, como ransomware, ataques de negação de serviço (DDoS) ou violações de dados, podem causar grandes danos à infraestrutura e aos sistemas de uma organização. Um plano de disaster recovery para esse cenário deve incluir medidas como backups regulares, sistemas de detecção de intrusão, planos de resposta a incidentes e planos de recuperação de dados seguros.

**Resolução:** Ataques cibernéticos: Caso ocorra um ataque cibernético, como um ransomware ou uma violação de segurança, a solução pode incluir a implementação de medidas de segurança robustas, como firewalls, antivírus e sistemas de detecção de intrusões. Além disso, é fundamental ter backups offline (isolados da rede) para garantir a disponibilidade dos dados em caso de criptografia ou perda.

**CENÁRIO E RESOLUÇÃO 04:**

**Cenário:** Interrupções de rede: Falhas na rede de comunicação, como problemas de conectividade com a Internet ou falhas em roteadores e switches, podem afetar a capacidade de uma organização em fornecer serviços e se comunicar interna e externamente. Um plano de disaster recovery nesse cenário pode envolver a redundância de conexões de rede, a configuração de rotas de contingência ou o uso de provedores de serviços de Internet alternativos.

**Resolução:** Falhas de comunicação: Se houver uma falha nas redes de comunicação, como a queda dos sistemas de telefonia ou da Internet, é importante ter uma solução alternativa. Isso pode envolver a utilização de linhas de comunicação redundantes, como links de Internet de backup ou conexões via satélite, para manter a comunicação interna e externa.

**CENÁRIO E RESOLUÇÃO 05:**

**Cenário:** Erros humanos: Enganos ou erros cometidos por funcionários podem causar interrupções significativas nas operações de uma organização. Esses erros podem variar desde a exclusão acidental de dados críticos até a instalação incorreta de atualizações de software. Um plano de disaster recovery deve abordar a educação e a conscientização dos funcionários, bem como procedimentos de backup e restauração de dados.

**Resolução:** Desastres físicos nas instalações: Em caso de incêndio, inundação, vandalismo ou outras situações que afetem as instalações físicas, a solução pode envolver a transferência temporária das operações para uma localização alternativa. Isso pode incluir a utilização de escritórios de recuperação de desastres ou espaços de trabalho temporários para garantir a continuidade das operações.

**RESUMO GERAL:**

Em todos esses cenários, um plano de disaster recovery eficaz envolve a implementação de medidas preventivas, como backups regulares, redundância de sistemas críticos, treinamento de funcionários e testes regulares do plano para garantir sua eficácia.

É importante ressaltar que cada empresa possui necessidades específicas de recuperação de desastres, e as soluções devem ser adaptadas de acordo com os requisitos, recursos e objetivos de negócios individuais. É recomendável envolver profissionais especializados em recuperação de desastres.

**Nome dos alunos do grupo:**

* Lucas dos santos Quintela
* Brian Lucas Chumacero Vanderlei
* Bruno Roberto Florentino da Silva
* Edivania de Albuquerque Moreira
* Bárbara Gotardi
* Lucas Gabriel