PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Arquitetura de Software Distribuído

Renan Aparecido Stuchi

SISTEMA PARA AQUISIÇÃO DE CUPOM PROMOCIONAL

Belo Horizonte
2019

Renan Aparecido Stuchi

SISTEMA PARA AQUISIÇÃO DE CUPOM PROMOCIONAL

Trabalho de Conclusão de Curso de Especialização em Arquitetura de Software Distribuído como requisito parcial à obtenção do título de especialista.

Orientador(a): Tadeu Faria

Belo Horizonte
2019

A minha família, as pessoas da minha vida e todos os professores do curso, que foram tão importantes na minha vida acadêmica e no desenvolvimento dessa monografia.

AGRADECIMENTOS

A Deus pela minha vida, pela minha família e pelas pessoas de minha vida.

A esta universidade, seu corpo docente, direção e administração, pela oportunidade de fazer essa tão importante especialização.

Agradeço a todos os professores por me proporcionar esse conhecimento, por tanto que se dedicaram, não somente por terem me ensinado, mas por terem me feito aprender a aprender.

Aos meus pais e irmão, pelo amor, incentivo e apoio incondicional.

Às pessoas de minha vida por direta ou indiretamente, nos bastidores, estarem me iluminando e clareando meus passos, sempre com muito amor, o meu muito obrigado de coração.

RESUMO

Esse projeto aborda uma solução de aquisição de cupom promocional, por meio da Internet, utilizando dispositivos móveis ou desktop. Essa solução se destina a todo consumidor final, ao comércio varejista e atacadista. Um cupom promocional é um artefato digital que os comerciantes podem oferecer de forma limita aos consumidores finais como o objetivo de oferecer descontos superiores aos produtos já ofertados em promoção.

O projeto visa substituir, em grande medida, a forma tradicional de produção e consumo de promoções, os panfletos promocionais, viabilizando a criação e distribuição dessas promoções em vias digitais. Pretende-se com essa abordagem facilitar a criação e a entrega das promoções aos consumidores finais, sendo o processo de criação das promoções gratuito e somente os produtos visualizados em cada entrega serão monetizados. À vista disso, a solução ambiciona liquidar os custos de produção e entrega das promoções tornando a monetização dessas mais justas para o comerciante varejista ou atacadista, pois só pagarão por seus produtos que realmente foram visualizados pelos consumidores, tornando esses custos uma fração muito pequena se comparado com o processo de produção, impressão e entrega dos panfletos. Outro ponto muito positivo em relação a essa solução é que ela reduz o desperdiço de recursos ambientais, como o papel, e isso não só impacta na abordagem de marketing da solução com também ajuda a manter as cidades mais limpas.

Sumariamente a solução proporcionará um desempenho extremamente elevado na criação e distribuição das promoções e cupons digitais, com custo extremamente baixos, alto índice de assertividade quando de posse dos produtos “favoritados” pelos consumidores finais de forma anônima e ganhos ambientais com a redução do consumo de papel, mantendo as cidades mais limpas.

O projeto está estruturado da seguinte forma: definição geral da solução, definição conceitual da solução, modelagem e projeto arquitetural, prova de conceito funcional e avaliação da arquitetura. Espera-se ao final do projeto que as integrações sejam atendidas com a prova de conceito proveniente das modelagens/definições deste documento.

Palavras-chave: arquitetura de software; projeto de software; requisitos arquiteturais; panfletos promocionais; consumidor final; comércio varejista e atacadista; cupom.

SUMÁRIO

1. Objetivos do trabalho

O objetivo geral deste projeto é apresentar uma proposta arquitetural de um sistema de aquisição de cupom promocional, por meio da Internet, utilizando dispositivos móveis ou desktop, visando substituir a forma tradicional de produção e consumo de promoções, os panfletos promocionais, viabilizando a criação, distribuição e aquisição dessas promoções em vias digitais.

Os objetivos específicos são:

1. Descrever os problemas e dificuldades que as promoções tradicionais, em forma de panfleto, do comércio varejista e atacadista sofrem e dar uma visão geral de uma solução que possibilite a criação, visualização, aquisição e acompanhamento de cupons promocionais digitais, com o custo e o tempo de criação e entrega dessas promoções extremamente reduzidas.
2. Expor a definição conceitual da solução, definindo os requisitos funcionais e não funcionais, as restrições arquiteturais e os mecanismos arquiteturais utilizados na solução.
3. Apresentar a modelagem arquitetural da solução, seus casos de uso com suas descrições resumidas, os diagramas de componente e implantação.
4. Evidenciar a prova de conceito, detalhar suas implementações e interfaces / APIs.
5. Fazer uma avaliação arquitetural da solução proposta, identificando seus atributos de qualidade, os cenários propostos para cada atributo de qualidade, avaliar o atendimento aos atributos de qualidade apresentando os resultados e a conclusão da proposta arquitetural.

2. Descrição geral da solução

2.1. Apresentação do problema

Apresentar seus produtos em promoção, aos consumidores finais, é uma necessidade básica do comércio varejista e atacadista. E esse processo fundamental ainda é arcaico, caro, ineficiente e demorado, pois utilizam, em grande medida, folhetos impressos com o intuito de informar os consumidores finais.

O processo tradicional é arcaico, pois utiliza papel, um recurso ambiental que deve ser utilizado e descartado com consciência, ou propagandas de rua com alto-falantes. O problema com os folhetos é que eles, muitas vezes são maus descartados, tornando as cidades sujas e com isso colaborando para outros problemas ambientais. Já as propagandas com alto-falantes já são proibidas em muitos centros urbanos devido à poluição sonora.

O processo tradicional é caro, pois é necessário pagar pela arte do panfleto, ou seja, seu desenvolvimento artístico, é necessário a plotagem, que pode ser muito cara à depender da qualidade e é necessário pagar pela entrega ou distribuição dos panfletos ao consumidor final.

O processo tradicional é ineficiente, pois na distribuição os panfletos podem chegar a mãos de consumidores finais que não estão interessados neles e assim serem descartados sem proveito algum. Ou seja, não há formas claras de saber o quão eficiente essa promoção foi, uma vez que não se sabe quantas pessoas se interessaram ou viram os produtos.

O processo tradicional é demorado devido ao tempo relativamente alto de produção e entrega dos panfletos aos consumidores finais.

Por o processo ser caro, demorado e não ter um índice de eficiência aceitável, muitos pequenos comércios deixam de fazê-lo. Outro problema com esse processo tradicional é quando o comerciante tem poucos ou apenas um produto para ofertar, assim os custos do processo promocional podem superar os ganhos que ele teria com o aumento do numero de venda desse produto, ficando ainda mais evidente se a quantidade do produto em estoque for baixa. Um caso típico desse cenário é a promoção de produtos que estão próximo ao vencimento, onde normalmente se tem produtos com vencimentos distintos e estoque reduzido.

A alternativa proposta para solucionar esses problemas está baseada no advento da Internet e do avanço/popularização dessa infraestrutura. Outro fator muito favorável a essa solução é o avanço dos dispositivos móveis, que tornam o acesso à informação muito mais rápido e abrangente. A solução proposta anseia ser a ponte que conecta os comerciantes

varejistas e atacadistas com seus consumidores finais, formando um elo forte e vitorioso para ambos os lados, pois os dois lados só têm a ganhar.

2.2. Descrição geral do software (Escopo)

O sistema propõe-se a melhorar a necessidade básica do comércio varejista e atacadista, no que tange seu processo fundamental de criar e distribuir promoções aos consumidores finais.

Para resolver os problemas dos custos e demora do processo de produção e distribuição dos folhetos promocionais, e também da utilização de recursos naturais (papel), o sistema torna fácil à criação de folhetos e cupons em formas digitais e sua distribuição nessas mesmas vias.

A solução utiliza a plataforma web para a criação e distribuição das promoções e cupons em vias digitais. O acesso ao sistema é publico e sua utilização apenas depende da simples criação de um dos perfis de acesso. Os dois perfis inicias do sistema são:

- **Comerciantes:** poderá criar promoções (lista de produtos em promoção e itens com cupons), limitar a quantidade de entrega dessa promoção aos consumidores finais, se for conveniente, definir o range de validade da promoção e acompanhar a visualização/procura de cada produto na lista. Esse perfil também pode realizar as ações presentes no perfil consumidor.

- **Consumidores:** poderá visualizar produtos em promoção, adquirir cupons, “favoritar” produtos, acompanhar seus cupons adquiridos (verificar se os produtos referentes a esses cupons foram separados em estoque e assim prontos para serem validados) e validar esses cupons junto ao comerciante no momento da compra.

Não é permita a compra do produto pelo sistema, tão pouco requisitar sua entrega. O propósito dessa solução é viabilizar as promoções entre os comerciantes e consumidores locais, tornando desapropriado os tramites e procedimentos de compra e transporte desses produtos.

3. Definição conceitual da solução

3.1. Requisitos Funcionais

A solução utiliza a arquitetura de microsserviços. Abaixo seguem os microsserviços e suas respectivas funcionalidades:

Microservice Register

- Cadastro de usuário

  O sistema deve permitir o cadastro de um novo usuário com o perfil Comerciante.

  O sistema deve permitir o cadastro de um novo usuário com o perfil Consumidor.

Microservice PyGoogleImg

- Busca de imagens de produtos

  O sistema deve possibilitar a busca por imagens sugeridas no Google Images, a partir do nome do produto informado ao cadastrar um produto em uma promoção.

Microservice Offers

- Cadastro de Oferta

  O sistema deve permitir aos usuários com perfil Comerciante a realização de cadastro de suas promoções (lista de produtos em promoção e produtos com cupons). O sistema deve possibilitar também limitar a quantidade de entrega dessa promoção aos consumidores finais, se for conveniente e definir o range de validade da promoção.

- Requisição de cupom

  O sistema deve permitir aos usuários com perfil Consumidor a requisição de cupom em produtos ofertados nessa modalidade.

- Acompanhamento de cupom

  O sistema deve permitir a ambos os perfis acompanhar quais produtos oferecidos na modalidade de cupom foram requisitados e em qual status eles se encontram (estoque finalizado, separado em estoque e entregue). O sistema

também deve permitir aos usuários do perfil comerciante na mudança no status de seus produtos oferecidos na modalidade de cupom.

Obs.: O status "estoque finalizado" é utilizado por comerciantes que não separam o estoque dos produtos oferecido como cupom e os deixam na estante de vendas, junto com os demais. Essa possibilidade pode entrecorrer em maus reviews ao comércio em questão.

Microservice Reviews

- Cadastro de reviews do produto

  O sistema deve permitir aos usuários do perfil consumidor cadastrar reviews aos produtos ofertados pelos comerciantes.

- Cadastro de reviews do comércio

  O sistema deve permitir aos usuários do perfil consumidor cadastrar reviews ao comércio autor de itens promocionais.

- Cadastro de reviews do consumidor

  O sistema deve permitir aos usuários do perfil comerciante cadastrar reviews referente ao consumidor final.

Microservice Favorites

- Cadastro de produtos favoritos

  O sistema deve possibilitar a inclusão de produtos e comércios à sua lista de favoritos para que o sistema possa destacar esses itens e/ou comércio "favoritado".

3.2 Requisitos Não-Funcionais

A seguir são apresentados os requisitos não funcionais do sistema:

- Usabilidade – O sistema deve prover boa usabilidade

Estímulo	Usuário realizando a criação de uma oferta.
Fonte do Estímulo	Usuário acessando a funcionalidade de criação de oferta.
Ambiente	Funcionamento, carga normal.

Artefato	Camada do cliente (*Web.Client*) e negócio (*Microservice Offers* e *PyGoogleImg*).
Resposta	A camada de apresentação denota facilidade de navegação, simplicidade de uso e objetividade nas informações expostas.
Medida da resposta	Usuário conseguiu ofertar um produto em no máximo 4 minutos.

- Acessibilidade – O sistema deve suportar ambientes Web responsivos e ambientes móveis.

Estímulo	Qualquer interface gráfica do usuário no sistema.
Fonte do Estímulo	Usuário acessando uma funcionalidade por meio de interface gráfica Web.
Ambiente	Funcionamento, carga normal.
Artefato	Camada do cliente (*Web.Client*).
Resposta	A camada de apresentação Web se adaptou as resoluções e tamanho das telas, mudando os componentes de posição de forma a melhorar a apresentação das informações e navegabilidade do usuário.
Medida da resposta	Identidade visual semelhante em todas as resoluções, componentes gráficos redimensionados de acordo com a resolução e tamanho.

- Desempenho – O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela inicial – exibição de ofertas).

Estímulo	Acessando a tela inicial do sistema (exibição/consulta de ofertas).
Fonte do Estímulo	Usuário acessando ou consultando ofertas na tela inicial do sistema.

Ambiente	Funcionamento, carga normal.
Artefato	Camada do cliente (*Web.Client*) e camada de acesso a dados no cliente (*javascript database*).
Resposta	O sistema respondeu com a totalidade das ofertas ou as ofertas que atendem aos filtros informados.
Medida da resposta	O sistema deve responder no caso mais dispendioso (totalidade dos registros) em menos de 2 segundos.

- Desempenho – O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela monitoramento de cupons).

Estímulo	Acessando a tela de cupons (monitoramento/consulta de cupons).
Fonte do Estímulo	Usuário acessando ou consultando cupons na tela de monitoramento de cupons do sistema.
Ambiente	Funcionamento, carga normal.
Artefato	Camada do cliente (*Web.Client*) e camada de acesso a dados no cliente (*javascript database*).
Resposta	O sistema respondeu com a totalidade dos cupons ou os cupons que atendem aos filtros informados.
Medida da resposta	O sistema deve responder no caso mais dispendioso (totalidade dos registros) em menos de 2 segundos.

- Manutenibilidade – O sistema deve apresentar manutenção facilitada.

Estímulo	Componentes de negócio do sistema que exigem sincronização com as bases de dados locais dos usuários (*javascript database*) precisam ser atualizados.
Fonte do Estímulo	Timeout durante a execução de um comando que exige sincronização com as base de dados locais dos usuários.

Ambiente	Indisponibilidade/Intermitência durante execução de comandos nos componentes de negócio em janela de atualização desses componentes (*schedule maintenance windows*).
Artefato	Componentes de negócio do sistema que exigem sincronização com as bases de dados locais.
Resposta	Todos os comandos com retorno de erro ou timeout devem ser reexecutados até que tenham retorno de execução com sucesso.
Medida da resposta	Comandos com erro ou timeout reexecutados e dados sincronizados nas base de dados locais dos usuários, sem que haja qualquer perca ou duplicidade dos dados (idempotência).

- Testabilidade – O sistema deve ser simples para testar

Estímulo	Execução de testes automatizados no sistema.
Fonte do Estímulo	Analista desenvolvedor.
Ambiente	Ambiente de desenvolvimento ou *staging* áreas em ambientes de homologação e produção.
Artefato	Componentes de negócio do sistema e arquiteturais do sistema.
Resposta	O sistema executa todos os testes automatizados em ambiente de desenvolvimento ou *staging* áreas em ambientes de homologação e produção.
Medida da resposta	O sistema deve possibilitar efetuar os testes com scripts automatizados executando com apenas um comando.

- Interoperabilidade – O sistema deve se comunicar com os sistemas dos fornecedores e um sistema interno de *Business Intelligence*. Alguns desses sistemas são antigos e desenvolvidos com tecnologia COBOL/CICS.

Estímulo	Mudanças (*change event)* na base de dados de oferta *Offers* ou nas bases de dados remotas de cada usuário.
Fonte do Estímulo	Comandos de negócio disparados pelos usuários que exijam alteração na base de dados de oferta *Offers* ou nas bases de dados remotas de cada usuário.
Ambiente	Sistema em funcionamento com carga normal.
Artefato	Base de dados de oferta *Offers* ou bases de dados remotas de cada usuário.
Resposta	O sistema deve publicar os eventos de alteração da base de dados *Offers* ao sistema de *BI* e os eventos de alteração das bases de dados remotas de cada usuário para os sistemas de cada respectivo fornecedore (comerciante) seguindo o padrão do *W3C* para *Server-Sent Events (Event Sourcing - https://www.w3.org/TR/eventsource/)*.
Medida da resposta	O sistema deve ser capaz de comunicar as alterações na base de dados de oferta *Offers* ou nas bases de dados remotas de cada usuário (*Changes Feeds)*, usando uma das abordagens técnicas de *Server-Sent Events: Polling, Long Polling e Event Source*.

- Disponibilidade – O sistema deve operar em qualquer período do dia e da noite.

Estímulo	Indisponibilidade em algum Nó do cluster de banco de dados ou componentes de negócio (*mircoservices*).
Fonte do Estímulo	Manutenção em algum nó do cluster (*schedule maintenance windows*) ou atualização dos componentes de negócio e/ou arquiteturais.
Ambiente	Diversos usuários estão utilizando o sistema.

Artefato	Base de dados ou componentes de negócio (*mircoservices*).
Resposta	Todos os usuários logados na aplicação devem continuar utilizando o sistema sem perceber que houve uma queda de um dos Nós do servidor de aplicação.
Medida da resposta	Todas as solicitações dos usuários devem ser atendidas, podendo haver um atraso de 3 segundos devido à queda de um dos Nós.

- Resiliência (*network*) – O sistema deve permanecer operando em condições adversas de conectividade.

Estímulo	Indisponibilidade de conectividade da aplicação cliente com os servidores (*back-end*).
Fonte do Estímulo	Falhas de conectividade com a internet ou ausência de área de cobertura.
Ambiente	Usuário utilizando o sistema com carga normal.
Artefato	Camada do cliente (*Web.Client*) e camada de acesso a dados no cliente (*javascript database*).
Resposta	O usuário logado na aplicação deve continuar utilizando o sistema sem perceber que houve uma indisponibilidade da rede de dados.
Medida da resposta	Todas as solicitações dos usuários relativas às consultas no sistema (*querys*) devem ser atendidas normalmente, não havendo impacto nessas funcionalidades centrais. As solicitações de comandos (*commands*) devem ser enfileiradas no banco de dados local do usuário para posterior sincronização.

- Segurança (*database per user*) – O sistema deve apresentar altos padrões de segurança.

Estímulo	Acessar o sistema sem estar logado.
Fonte do Estímulo	Usuários com perfil Comerciante ou Consumidor.
Ambiente	Usuário utilizando o sistema com carga normal.
Artefato	Todos os componentes do sistema, exceto o componente de cadastro de usuário (*microservice Register*).
Resposta	O sistema não será capaz de sincronizar a base de dados local do usuário, o que impossibilita o recebimento de novas ofertas e execuções de comandos no sistema.
Medida da resposta	O sistema não deve permitir a sincronização da base de dados local do usuário com sua base de dados remota, caso este usuário já tenha se cadastro anteriormente no sistema.

- Escalabilidade – O sistema deve ser capaz de escalar vertical e horizontalmente no cluster, de forma manual ou automaticamente.

Estímulo	Crescimento da utilização de recursos do sistema por aumento da quantidade média de usuários simultâneos ou por picos de utilização do sistema.
Fonte do Estímulo	Usuários com perfil Comerciante ou Consumidor.
Ambiente	Usuário utilizando o sistema com carga média acima da normal ou em momentos de pico, ocasionado por um aumento rápido da quantidade de usuários simultâneos no sistema.
Artefato	Todos os componentes do sistema em *back-end* (*microservices e database*).
Resposta	O sistema deverá ser capaz de fazer escalonamento vertical e horizontal no *cluster* de *microservices* por meio do orquestrador de *microservices* e do cluster do banco de dados por meio do sistema gerenciado de banco de dados distribuído.
Medida da resposta	O sistema deve escalar vertical e horizontalmente os *microservices* de forma elástica dentro do *cluster* de

	microservices e as instâncias de banco de dados devem ser escaladas de forma crescente apenas, de acordo com os indicadores de desempenho (*key performance indicator*) configurados.

3.3. Restrições Arquiteturais

- O sistema deve ser desenvolvido em .Net Core 2, porém pode haver *microservices* que podem ser melhor executados em outras tecnologias que mantenham interoperabilidade com o .Net Core 2.
- Os componentes arquiteturais devem ser desenvolvidos e/ou selecionados para execução em cluster facilitando a escalabilidade da aplicação
- O sistema deve facilitar a comunicação com os demais sistemas adotando formas de comunicação padronizadas por entidades como o W3C.
- O sistema deve garantir que cada usuário tem seu próprio banco de dados, garantindo melhor segurança e isolamento.
- O sistema deve permitir sua utilização em modo *offline.*
- O sistema deve abrir de forma responsiva em aparelhos menores, como celular e *tablet*.
- O aplicação cliente deve utilizar os novos recursos do *webassembly (https://webassembly.org/)* para explorar os limites do javascript no browser, como a criação de *threads* para aumentar a performance dessas aplicações, se utilizando de outras linguagens, por exemplo, C, C++, Rust, C# etc.
- A aplicação cliente não recarregar páginas dinâmicas do servidor, elas devem ser redesenhadas conforme a interação do usuário localmente no browser (*single page application*).

3.4. Mecanismos Arquiteturais

Mecanismo de Análise	**Mecanismo de Design**	**Mecanismo de Implementação**
Persistência back-end	NoSQL	CouchDB
Persistência front-end	NoSQL	PouchDB

SAAS database architecture	*One database per user* | *CQRS*	CouchDB – couch_peruser option
Comunição entre processos (*Inter Process Communication*)	EventSource (Event Driven) e RESTful	CouchDB Changes Feeds e ASP.NET Core 2 Web API
Comunicação entre objetos (*Inter Object Communication*)	Mediator	MediatR
Interação entre microservices	Orchestration	Microsoft Service Fabric
Monitoração do cluster de microservices	Monitoring (DevOps)	Microsoft Service Fabric Explorer
Changes Feeds Observable	Server-Sent Event (EventSource)	CouchDB Speigel
Mensageria	Dumb Broker	Kafka
Isolamento de processos	Containers	Docker
Log	Framework de Log	Log4Net
Build	Compilação	MSBuild/Nuget
Deploy	Deploy da aplicação no cluster	VSTS/ Microsoft Service Fabric Deployment
Front-End	WebAssembly	Blazor
Versionamento	Source Control Mgmt.	Git
Autenticação e Autorização	Basic Authentication e Cookie Authentication	CouchDB Basic authentication (RFC 2617) e Cookie authentication (RFC 2109)
Independência de implementação da aplicação cliente e servidor.	noBackend	PouchDB, CouchDB e CQRS
Layout responsivo	Html flexbox e media queries.	Bootstrap 4
Alta Disponibilidade	Cluster	Cluster CouchDB, Cluster Spiegel

(*High-Availability*)		e Cluster Service Fabric

4. Modelagem e projeto arquitetural

4.1. Modelo de casos de uso

O diagrama de casos de uso oferece uma visão global dos casos de uso e dos atores que dele participam. Para uma melhor analise arquitetural do projeto, separei os casos de uso compreendidos por cada *microservices*.

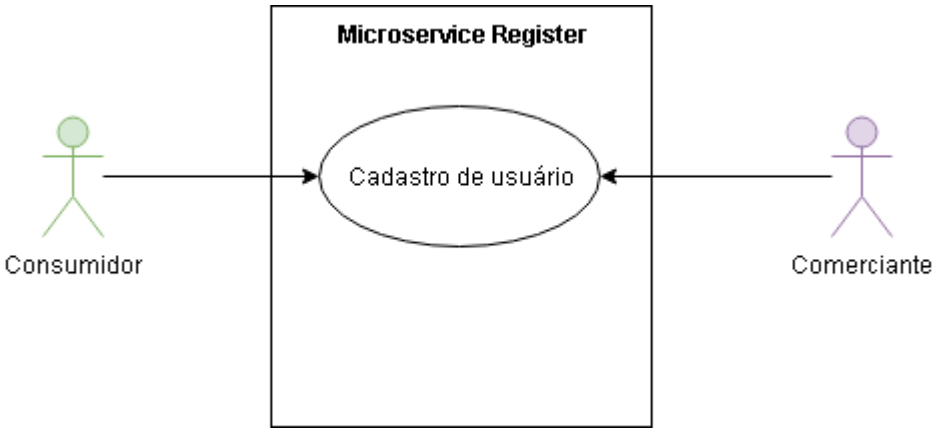


Figura 1 - Diagrama de Casos de Uso: Microservice Register

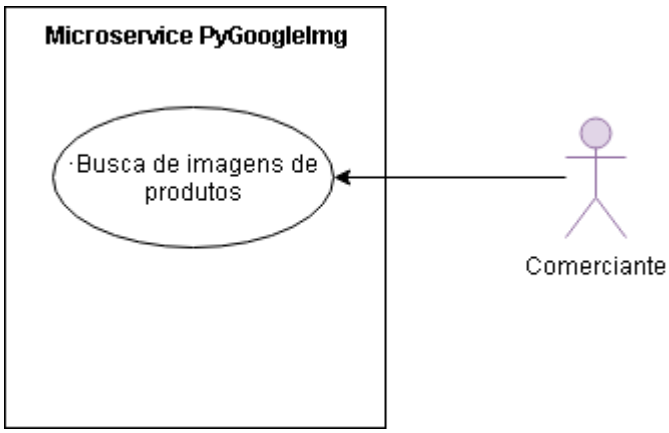


Figura 2 - Diagrama de Caso de Uso: Microservice PyGoogleImg

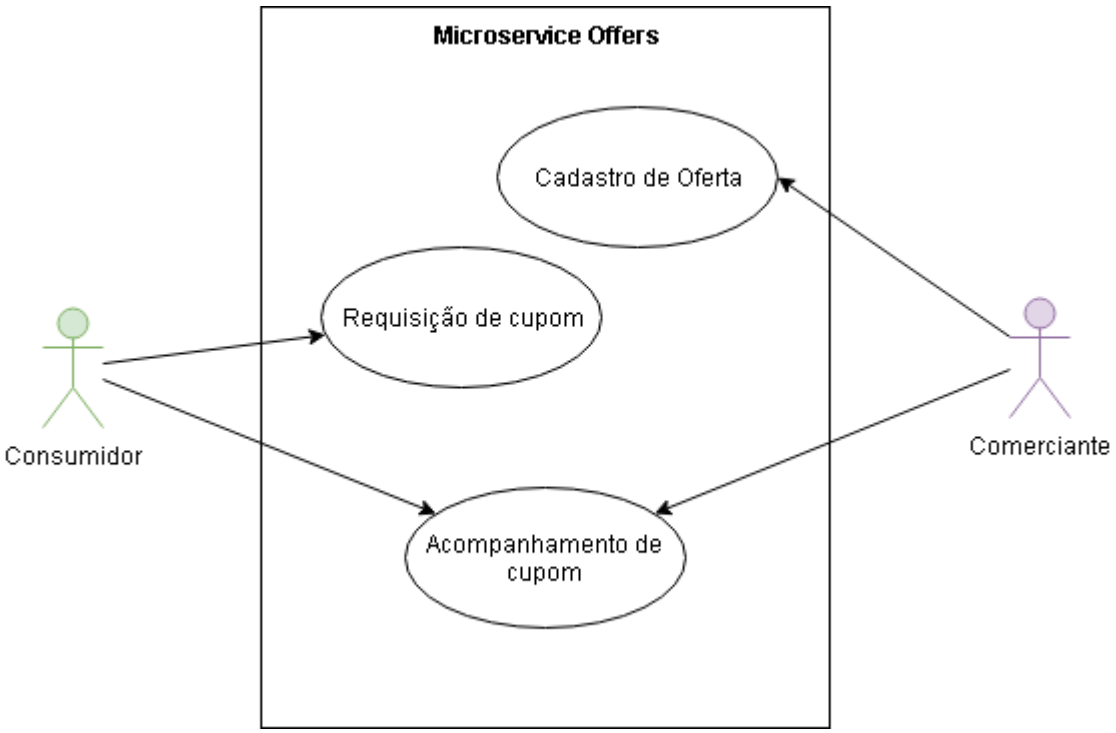


Figura 3 - Diagrama de Caso de Uso: Microservice Offers

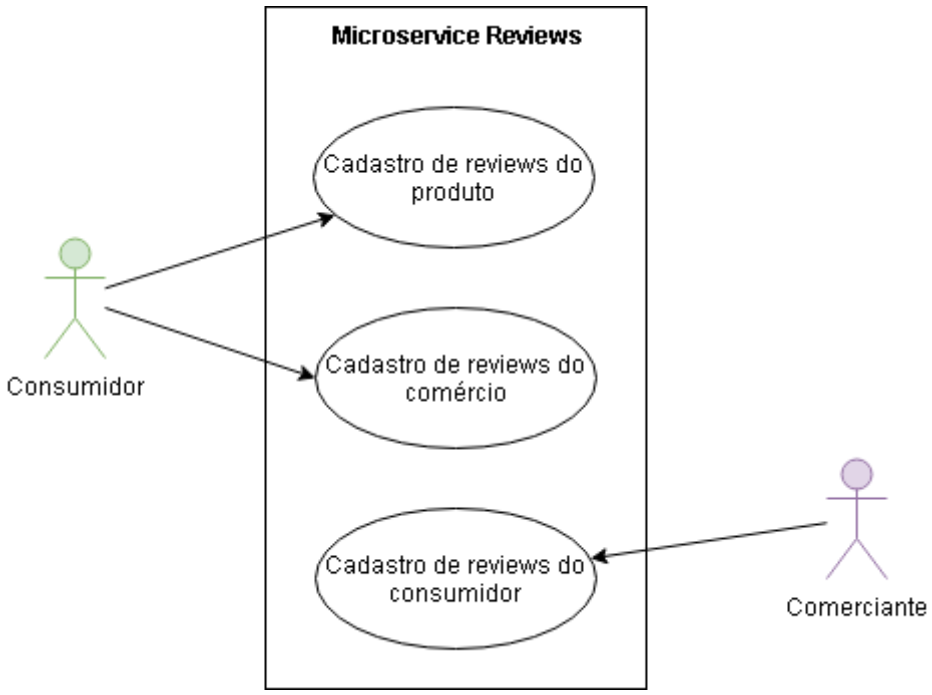


Figura 4 - Diagrama de Caso de Uso: Microservice Reviews

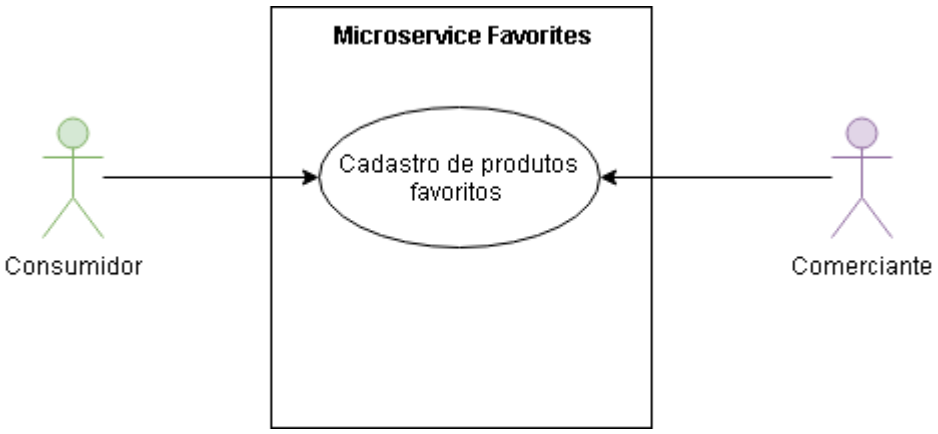


Figura 5 - Diagrama de Caso de Uso: Microservice Favorites

4.2. Descrição resumida dos casos de uso

Caso de uso: Cadastro de usuário

ID	1
Estória do usuário	Cadastrar usuário
Ator	Como um comerciante ou um consumidor final
Descrição	Eu quero ser capaz de cadastrar um usuário no sistema, por meio da Web ou dispositivos móveis, acessando a página de *login* do sistema e em seguida o *link* de cadastro de novo usuário. O sistema deve apresentar um formulário no qual o usuário pode informar seu nome, e-mail, senha, se este é comerciante e caso seja o nome do comércio.
Valor do negócio	Para que seja possível a inclusão de comerciantes e consumidores finais no sistema.
Prioridade	Alta.
Estimativa	8

Caso de uso: Busca de imagens de produtos

ID	2
Estória do usuário	Buscar imagens de produtos
Ator	Como um comerciante
Descrição	Eu quero ser capaz de cadastrar meus produtos em promoção, por meio da Web ou dispositivos móveis, acessando a página de adição de ofertas, no menu, presente apenas para o perfil comerciante. O sistema deve apresentar um formulário no qual o usuário comerciante pode informar entre outras informações o nome e preço de seus produtos nesta oferta, se estes produtos terão cupons e a quantidade de cupons. Assim que o nome do produto for definido em seu respectivo campo no formulário, o sistema deve dar sugestões de imagens desse produto vindas do Google Imagens, por meio de *scraping* da página do buscador.
Valor do negócio	Para que seja possível a inclusão de imagens aos produtos em promoção oferecidos pelos comerciantes no sistema.
Prioridade	Baixa.
Estimativa	5

Caso de uso: Cadastro de ofertas

ID	3
Estória do usuário	Cadastrar oferta e seus produtos
Ator	Como um comerciante
Descrição	Eu quero ser capaz de cadastrar meus produtos em promoção, por meio da Web ou dispositivos móveis, acessando a página de adição de ofertas, no menu do sistema,

	presente apenas para o perfil comerciante. O sistema deve apresentar um formulário no qual o usuário comerciante pode o nome da promoção, um intervalo de datas de inicio e fim das entregas da promoção pelo sistema, o nome e preço de seus produtos nesta oferta, se estes produtos terão cupons e a quantidade de cupons. O sistema deve possibilitar também limitar a quantidade de entregas dessa promoção aos consumidores finais, se for conveniente a essa promoção.
Valor do negócio	Para que seja possível a inclusão de ofertas e sues produtos em promoção pelos comerciantes do sistema.
Prioridade	Alta.
Estimativa	5

Caso de uso: Requisição de cupons

ID	4
Estória do usuário	Requisitar cupons
Ator	Como um comerciante ou um consumidor final
Descrição	Eu quero ser capaz de requisitar cupons de produtos oferecidos nesta modalidade (modalidade onde o comerciante coloca uma quantidade especifica de produtos com valores inferiores aos já ofertados em promoção para alavancar a saída do produto) no sistema, por meio da Web ou dispositivos móveis, acessando a página principal do sistema. O sistema deve apresentar, em sua página principal, a listagens dos produtos em ofertas. Os produtos podem ser filtrados por qualquer coluna de texto da listagem (como nome do produto, nome do comércio etc.) e ordenados por qualquer uma delas. Nos botões de ações, ultima coluna à direita da listagem, o sistema deve permitir clicar no

	botão de requisição de cupom dos produtos que são oferecidos nesta modalidade. Os comerciantes não serão capazes de requisitar cupons de seus próprios produtos, apenas produtos de outros comerciantes.
Valor do negócio	Para que seja possível requisitar cupons de produtos oferecidos nesta modalidade, por qualquer usuário no sistema.
Prioridade	Média.
Estimativa	3

Caso de uso: Acompanhamento de cupom

ID	5
Estória do usuário	Acompanhar cupons
Ator	Como um comerciante ou um consumidor final
Descrição	Eu quero ser capaz de acompanhar meus cupons requisitados no sistema, por meio da Web ou dispositivos móveis, acessando o item de menu acompanhamento de cupons, na página principal do sistema. O sistema deve apresentar uma listagem de cupons oferecidos em produtos promocionais da modalidade de cupom e uma listagem de cupons requisitados quando o perfil de acesso ao sistema for comerciante e apenas uma listagem de cupons requisitados quando o perfil de acesso ao sistema for consumidor. Os cupons podem ser filtrados por qualquer coluna de texto da listagem e ordenados por qualquer uma delas. Nessas listagens será possível acompanhar o status de cada cupom por usuário. Os status de um cupom são: cupom requisitado, cupom finalizado em estoque, cupom separado em estoque, cupom/produto entregue e cupom cancelado.
Valor do negócio	Para que seja possível acompanhar cupons de produtos oferecidos nesta modalidade, por

	qualquer usuário no sistema.
Prioridade	Média.
Estimativa	5

Caso de uso: Cadastro de reviews do produto

ID	6
Estória do usuário	Cadastrar reviews de produtos
Ator	Como um comerciante ou um consumidor final
Descrição	Eu quero ser capaz de cadastrar reviews de produtos no sistema, por meio da Web ou dispositivos móveis, acessando por meio do link na listagem de cada produto, na página inicial do sistema ou na página de acompanhamento de produtos. O sistema deve apresentar uma listagem com os reviews do produto e um formulário para o cadastramento de um novo review, caso esse usuário não tenha feito um review desse produto anteriormente. Caso o usuário já tenha feito um review para esse produto o sistema deverá permitir a edição desse review, carregando os dados no formulário para edição. No formulário o usuário poderá informar sua avaliação do produto entre uma e cinco estrelas, um título e um comentário para a review. Os comerciantes não serão capazes de cadastrar reviews de produtos seus próprios produtos promocionais, apenas em produtos de outros comerciantes.
Valor do negócio	Para que seja possível cadastrar reviews de produtos promocionais oferecidos pelo sistema, por qualquer usuário no sistema.
Prioridade	Média.
Estimativa	5

Caso de uso: Cadastro de reviews do comércio

ID	7
Estória do usuário	Cadastrar reviews do comércio
Ator	Como um comerciante ou um consumidor final
Descrição	Eu quero ser capaz de cadastrar reviews de comércios no sistema, por meio da Web ou dispositivos móveis, acessando por meio do link na listagem de cada produto oferecido pelo comércio, na página inicial do sistema ou na página de acompanhamento de produtos. O sistema deve apresentar uma listagem com os reviews do comércio e um formulário para o cadastramento de um novo review, caso esse usuário não tenha feito um review desse comércio anteriormente. Caso o usuário já tenha feito um review para esse comércio o sistema deverá permitir a edição desse review, carregando os dados no formulário para edição. No formulário o usuário poderá informar sua avaliação do comércio entre uma e cinco estrelas, um título e um comentário para a review. Os comerciantes não serão capazes de cadastrar reviews para seu próprio comércio, apenas para outros.
Valor do negócio	Para que seja possível cadastrar reviews de comércios que oferecem produtos promocionais no sistema, por qualquer usuário no sistema.
Prioridade	Média.
Estimativa	5

Caso de uso: Cadastro de reviews do consumidor

ID	8
Estória do usuário	Cadastrar reviews do consumidor

Ator	Como um comerciante
Descrição	Eu quero ser capaz de cadastrar reviews do consumidor no sistema, por meio da Web ou dispositivos móveis, acessando por meio do link na listagem de cada produto oferecido pelo comércio, requisitados pelo consumidor, na página de acompanhamento de cupons do sistema. O sistema deve apresentar uma listagem com os reviews do consumidor e um formulário para o cadastramento de um novo review, caso esse usuário não tenha feito um review desse consumidor anteriormente. Caso o usuário já tenha feito um review para esse comércio o sistema deverá permitir a edição desse review, carregando os dados no formulário para edição. No formulário o usuário poderá informar um título e um comentário para a review. Os comerciantes não serão capazes de cadastrar reviews para si próprio, como consumidores, apenas para outros.
Valor do negócio	Para que seja possível cadastrar reviews do consumidor que consomem produtos promocionais no sistema, para que os comerciantes possam ver.
Prioridade	Média.
Estimativa	5

Caso de uso: Cadastro de produtos favoritos

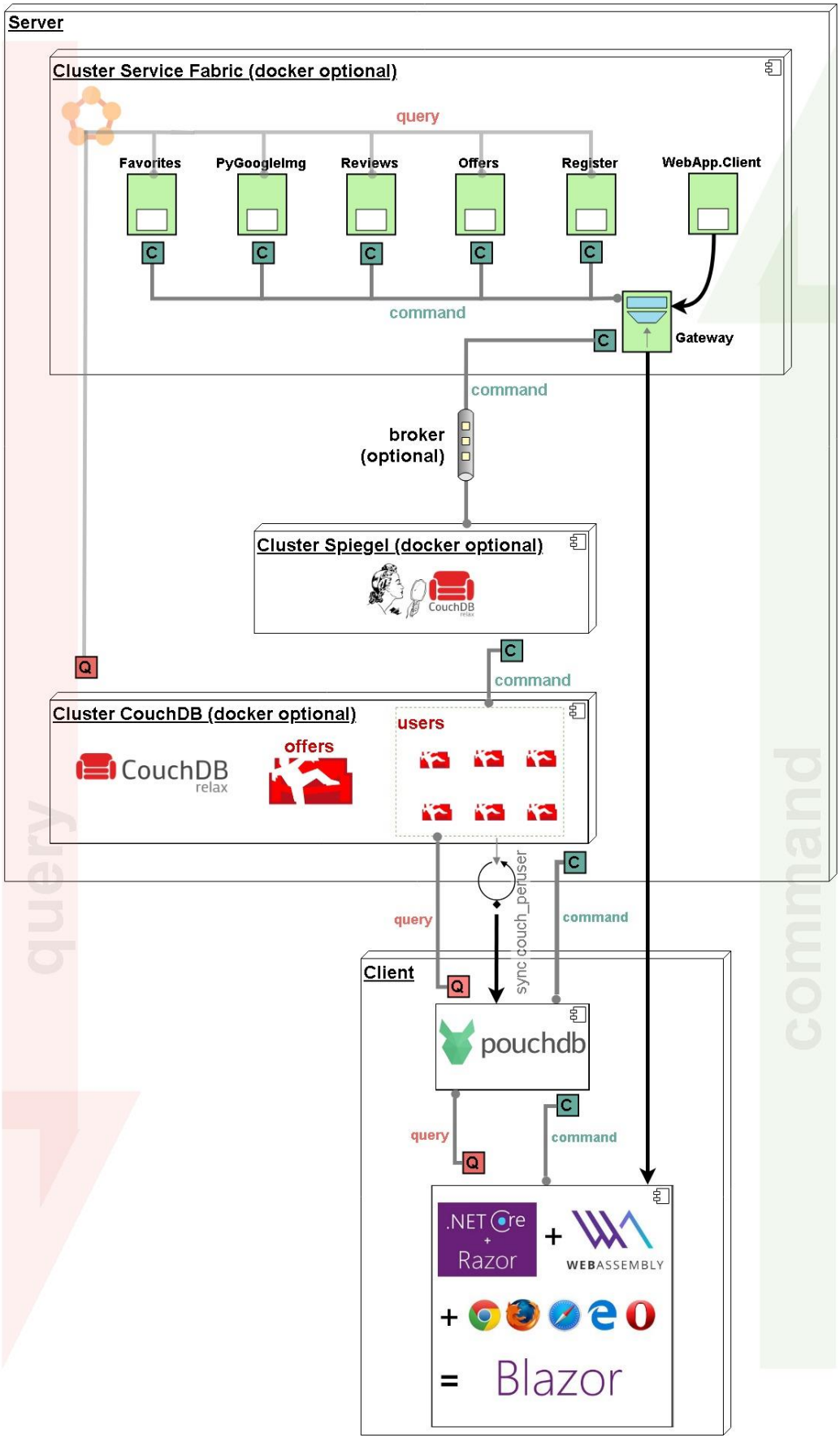
ID	9
Estória do usuário	Cadastrar produtos/comércios favoritos
Ator	Como um consumidor ou comerciante
Descrição	Eu quero ser capaz de cadastrar produtos como favoritos no sistema, por meio da Web ou dispositivos móveis, acessando por meio do link na listagem de cada produto, na página inicial do sistema, na página de acompanhamento de produtos ou localizando

	o produto ou comércio na tela de cadastro de favoritos. O sistema deve apresentar uma listagem com os produtos/comércios adicionados, podendo o usuário remover qualquer produto ou comércio presente na listagem de favoritos. Produtos ou comércios na lista de favoritos permitem que o sistema dispare notificação quando novas promoções contendo esses itens são criadas ou o comércio favorito crie novas ofertas. O usuário também poderá utilizar os favoritos para a criação de listagem de compras, utilizadas para cotação dos melhores preços promocionais encontrados no momento, podendo ser aplicado um filtro de comércios no resultado.
Valor do negócio	Para que seja possível cadastrar produtos e comércios favoritos no sistema, facilitando a notificação de novas promoções e a busca por listagem de compras.
Prioridade	Média.
Estimativa	5

Caso de uso: Sistema de atendimento ao consumidor (SAC)

Inicialmente o sistema de atendimento ao consumidor será mantido/atendido fora do sistema por utilização e-mail e atendido caso a caso prontamente. Posteriormente, esses chamados serão cadastros no sistema para que haja um estudo para a automatização do atendimento em chamados possíveis.

4.3. Modelo de componentes



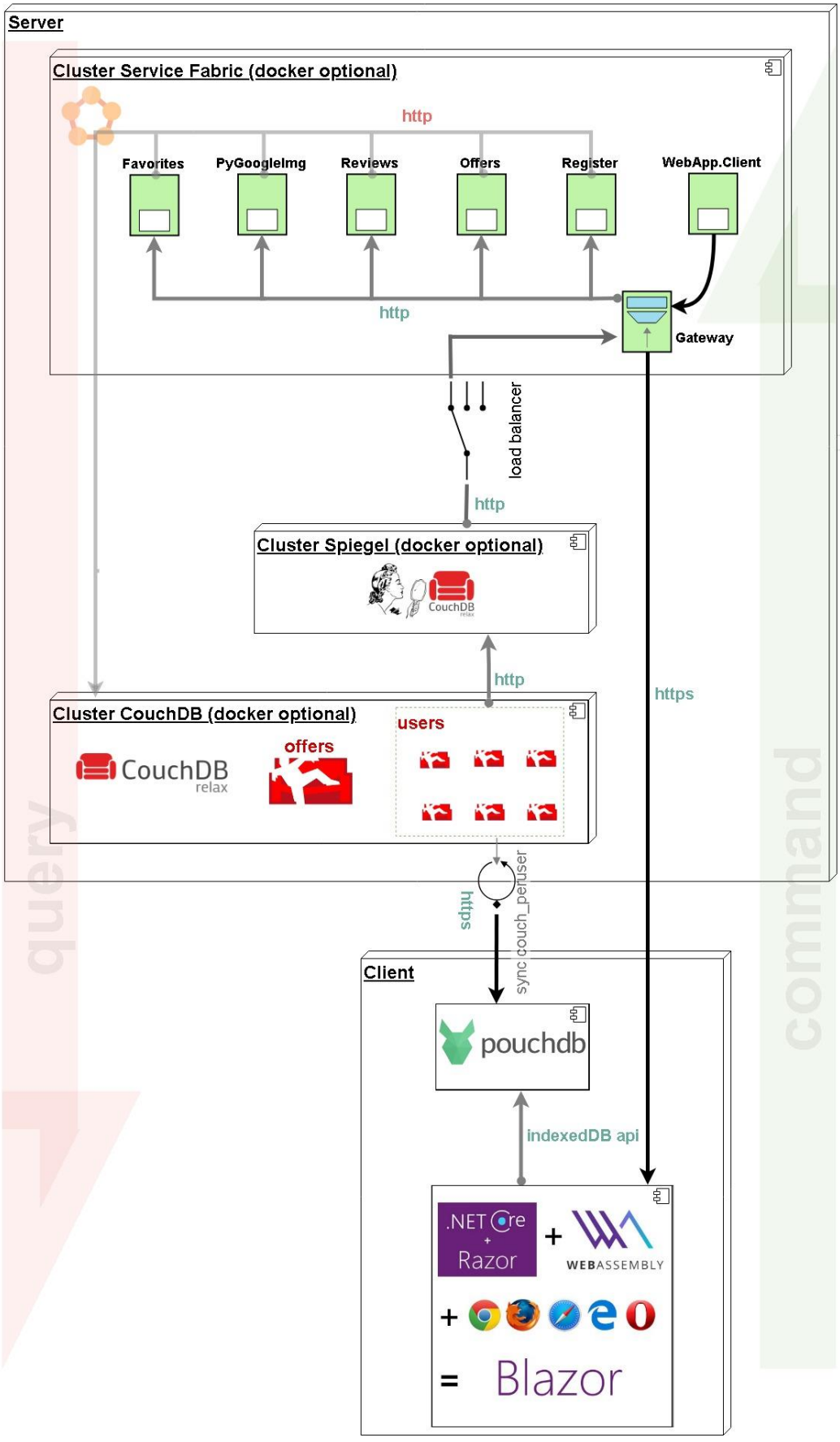
A seguir é mostrado um detalhamento dos componentes e os módulos/clusters envolvidos no modelo de componentes. Nessa arquitetura devemos considerar dois grandes blocos distintos, sendo essa divisão composta pelo cliente web/móbile, tendo seus principais componentes o Blazor e PouchDB, e o bloco servidor, tendo seus principais componentes o cluster CouchDB, o cluster Spiegel e o cluster Service Fabric. O *frontend* foi desenvolvido com a abordagem *noBackend* o que garante uma implementação totalmente independente do bloco cliente com o bloco servidor. A utilização das abordagens *noBackend,CQRS* e das múltiplas instancias do CouchDB, Spiegel e dos serviços do Service Fabric distribuidas em seus respectivos clusters, on-premise ou cloud, irão garantir/atestar os requisitos não a funcionais de alta disponibilidade (*HA:High-Availability*) e desempenho, não só pela distribuição de carga nos clusters, mas também pelo aspecto extremamente relevante que o *noBackend e o CQRS* trazem permitindo que as consultas/*querys* da aplicação cliente ocorram dentro do banco de dados local no próprio cliente, evitando essa carga no *backend* que se preocupa com a execução de comandos apenas. A atualização dos dados na base de dados local do cliente acontece por meio do processo sincronização entre o Couch (base de dados remota) e o Pouch (Base de dados local) no momento em que novos comandos são gerados no Pouch ou quando novos objetos de consulta são criados ou atualizados no Couch, à medida em que houver rede de comunicação é estabelecida ou restabelecida. O cluster do Service Fabric é construído e implantado por ferramentas de analise de código, integração e entrega continua, garantindo/atestando os requesitos não funcionais de manutenabilidade, testabilidade e código limpo. A atualização dos clusters do CouchDB e Spiegel não devem ocorrer frequentemente, uma vez que estes produtos sofrem atualizações frequentes. Já o cluster de serviços do Service Fabric compreende os *microservices* da aplicação e por isso sofrem constantes atualizações orquestradas pelo serviço de update do Service Fabric (Update-ServiceFabricService), o que garante o *Zero Downtime* da aplicação.

Componente	Descrição
Blazor	Componente WebAssembly que representa a aplicação cliente. Sua plataforma de execução pode ser o browser ou webview, pode executar códigos c#, c++, c, ou qualquer que suporta wasm. Sua vantagem é a execução imperativa de código compilado no navegador, com performance nativa e independente da interpretação do motor de renderização de cada navegador.
PouchDB	Implementação do CouchDB in javascript para utilização como banco de dados local em WebApp rodando no navegador, com capacidade de sincronização com sua contraparte CouchDB no servidor.
CouchDB	Banco de dados distribuído, capaz de gerenciar milhares de banco de dados e conexões por

	instancia, com capacidade de sincronização entre as instancias e o PouchDB. Uma de suas mais conhecidas abordagens de utilização é a couch_peruser, a qual cada usuário da aplicação tem seu banco de dados particular e é essa abordagem que fui utilizada nesta solução. A autenticação dos usuários é feita pelo CouchDB, o qual garante que cada usuário acesse apenas o seu banco de dados.
Spiegel	Componente distribuído responsável pelo envio de novos comandos, inseridos nas bases de dados dos usuários (CouchDB), para Api Gateway, a qual o enviará ao serviço responsável pelo processamento do evento/comando.
Gateway (Service Fabric)	Serviço seguro com múltiplas instancias distribuídas no cluster do service fabric, responsável por ser a porta de entrada de comandos no cluster de serviços. O serviço de Gateway recebe os comandos do Spiegel ou Kafka e direciona esses comandos ao serviço responsável pela execução do comando.
WebApp.Client (Service Fabric)	Assembly Blazor anexado ao serviço de Gateway que representa o aplicativo cliente. Esse assembly é enviado ao cliente ao requisitar o app.
Register (Service Fabric)	Serviço com múltiplas instancias distribuídas no cluster do service fabric, responsável por registrar novos usuários no sistema.
Offers (Service Fabric)	Serviço com múltiplas instancias distribuídas no cluster do service fabric, responsável pelos comandos de criação de ofertas e manipulação de seus produtos.
Reviews (Service Fabric)	Serviço com múltiplas instancias distribuídas no cluster do service fabric, responsável pelos comandos relacionados a criação e manutenção de reviews de produtos, comércios e consumidores.
PyGoogleImg (Service Fabric)	Serviço com múltiplas instancias distribuídas no cluster do service fabric, responsável pelo scraping python de imagens de produtos do Google Images.
Favorites (Service Fabric)	Serviço com múltiplas instancias distribuídas no cluster do service fabric, responsável pelos comandos relacionados a criação e manutenção de

	produtos/listas em favoritos.
Broker kafka (optional)	Componente opcional, porém recomendado para ambiente de desenvolvimento, teste e produção, que centralizará e armazenará todos os comandos dos usuários, simplificando o consumo, o gerenciamento e a divisão desses eventos por tipo.
Docker (optional)	Componente opcional, porém recomendado para ambiente de desenvolvimento, teste e produção, responsável pelo isolamento e virtualização dos componentes no cluster de maquinas físicas. CouchDB (kubernetes cluster orchestration): https://hub.docker.com/_/couchdb/ https://hub.kubeapps.com/charts/incubator/couchdb Spiegel (Swarm/kubernetes cluster orchestration): https://hub.docker.com/r/redgeoff/spiegel/ Kafka (kubernetes cluster): https://hub.docker.com/r/bitnami/kafka/ https://hub.kubeapps.com/charts/bitnami/kafka Services (Service Fabric orchestration): O orquestrador do service fabric pode trabalhar com *microservices* executáveis (.exe) e containers windows ou linux.

4.4. Modelo de implantação



A seguir é mostrado um detalhamento dos módulos/serviços envolvidos em cada nodo/cluster e como o modelo de implantação deve ser implementado para atender principalmente os requisitos não funcionais de disponibilidade e performance. No modelo acima não está explicito a arquitetura de cluster. Caso a opção de implantação seja com a utilização de containers docker os, os clusters não devem ser balanceados, uma vez que tecnologias de orquestração como Kubernetes e Swarm já contém lógicas internas para distribuição da carga e mesmo sem a utilização de docker e suas tecnologias de orquestração, os clusters CouchDB, Spiegel e Service Fabric são capazes distribuir suas cargas internamente, sem a utilização de um load balancer explicito. A utilização de um load balancer explicito para o serviço de entrado do Service Fabric, o Gateway, é opcional, porém recomendada, ainda que o Service Fabric reinicie seus serviços em caso de falha para que estes permaneçam sempre ativos, é um bom *design* manter varias instancias desse serviço para manter a disponibilidade visto que não há problemas de performance com este serviço, uma vez que não há processamento de comandos nesse serviço, funcionando com um proxy de entrada para o processamento dos comandos pelos demais serviços. A utilização das abordagens *noBackend,CQRS* e das múltiplas instancias do CouchDB, Spiegel e dos serviços do Service Fabric distribuidas em seus respectivos clusters, on-premise ou cloud, irão garantir/atestar os requisitos não a funcionais de alta disponibilidade (*HA:High-Availability*) e desempenho, não só pela distribuição de carga nos clusters, mas também pelo aspecto extremamente relevante que o *noBackend e o CQRS* trazem permitindo que as consultas/*querys* da aplicação cliente ocorram dentro do banco de dados local no próprio cliente, evitando essa carga no *backend* que se preocupa com a execução de comandos apenas. A atualização dos dados na base de dados local do cliente acontece por meio do processo sincronização entre o Couch (base de dados remota) e o Pouch (Base de dados local) no momento em que novos comandos são gerados no Pouch ou quando novos objetos de consulta são criados ou atualizados no Couch, à medida que houver rede de comunicação é estabelecida ou restabelecida.

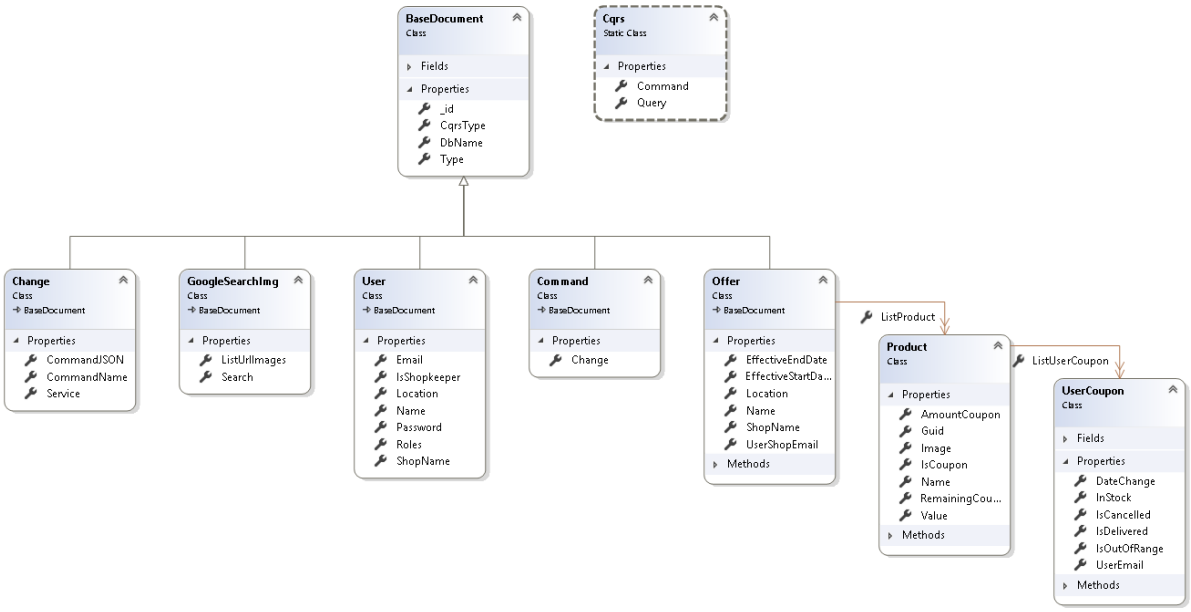
Nodo	Descrição
Client (Browser)	Qualquer dispositivo provido de web browser ou webview está apto a executar o aplicativo cliente, composto pelos componentes PouchDB e Blazor.
Cluster CouchDB	Cluster criado sobre as maquinas físicas ou virtuais (http://docs.couchdb.org/en/2.2.0/cluster/index.html) ou virtualizado sobre o kurbernetes com docker (https://hub.kubeapps.com/charts/incubator/couchdb). Automaticamente balanceado entre as instancias do CouchDB. Cluster que representa a base de dados distribuída, não relacional, do sistema. Implementada na abordagem “couch_peruser”, a qual garante um banco de dados remoto por usuário, facilitando a sincronização com a base de dados local do usuário,

	PouchDB (banco de dados javascript que implementa fiel e totalmente o protocolo de sincronização CouchDB).
Cluster Spiegel	Cluster criado sobre maquinas físicas ou virtuais (https://www.npmjs.com/package/spiegel) ou virtualizado sobre o Swarm ou Kubernetes (https://hub.docker.com/r/redgeoff/spiegel/). Automaticamente balanceado entre as instancias do *Spiegel Update Listener* e *Change Listener*. Cluster Spiegel para comunicação de novos comandos entregues às bases de dados remotas dos usuários, por meio do processo de sincronização do CouchDB com o PouchDB, ao *event store* Kafka ou diretamente o serviço Gateway do Service Fabric.
Cluster Kafka (optional)	Cluster opcional virtualizado sobre o Kubernetes (https://hub.kubeapps.com/charts/bitnami/kafka). Automaticamente balanceado entre os Pods Kubernetes. Esse cluster representa o cluster de *event store* da aplicação, um *Dumb Broker*, para aplicação avançada do estilo arquitetural de *Event Driven*.
Load Balancer (optional)	Load Balancer opcional, porém recomendado para balancear a carga do serviço de entrada do Service Fabric, o *Gateway Service*, caso esse serviço seja implantado em múltiplas instancias, para garantir a alta disponibilidade, uma vez que o Service Fabric reinicia suas instancias de serviços em caso de indisponibilidade.
Service Fabric Cluster	Cluster criado sobre maquinas físicas ou virtuais que pode trabalhar com *microservices* executáveis (.exe) e containers windows ou linux. Esse cluster representa o cluster das instancias dos microsserviços da aplicação.

4.5. Modelo de dados (opcional)

O diagrama de classes abaixo representa os documentos de usuário e oferta, mantidos pelo sistema de banco de dados distribuído CouchDB, nos bancos de dados _users, offers e

userdb_{hex encoded username}. Esse diagrama representa a parte do sistema tratado na POC.



5. Prova de conceito / protótipo arquitetural

5.1. Implementação e implantação

A prova de conceito desse projeto visa atestar a qualidade do sistema no que toca os principais requisitos não funcionais ou de qualidade, descritos na seção 3.2 deste documento, obedecendo as restrições arquiteturais impostas na seção 3.3, sendo considerado na implementação os casos de uso críticos. Com isso, tem-se a pretensão de minimizar riscos e maximizar ganhos de produtividade na sequência do projeto.

As tecnologias usadas na POC são as descritas na seção 3.4 de mecanismos arquiteturais.

As telas da aplicação podem ser vistas nas evidências da avaliação.

Os casos de uso utilizados na validação dessa POC são:

- **Cadastro de usuário**
- **Busca de imagens de produtos**
- **Cadastro de oferta**
- **Requisição de cupom**
- **Acompanhamento de cupom**

Nessa POC pretende-se validar os seguintes requisitos não funcionais:

- **Segurança (*database per user*)– O sistema deve apresentar altos padrões de segurança.**

  Esse RNF foi escolhido devido a preocupação em manter dados seguros e evitar falhas de segurança no projeto.

  Os critérios de aceite são:

  - Não permitir a sincronização entre a base de dados local do usuário com sua base de dados remota sem estar autenticado no sistema.

- Não permitir a conexão de um usuário autenticado a uma base de dados de outro usuário.

- **Acessibilidade – O sistema deve suportar ambientes Web responsivos e ambientes móveis.**

  Esse RNF foi escolhido para garantir que atenda todas as exigências da arquitetura em ter um sistema responsivo e que se adapte em celulares, tablets e desktops.

  Os critérios de aceite são:

  - A tela do sistema deve apresentar facilidade de navegação e os objetos da tela devem se adaptar de acordo com a resolução identificada, tanto para celulares como desktops.
  - O sistema deve se manter com o mesmo padrão de cores e objetos.
  - O sistema deve ser compatível com os principais browsers do mercado como: Edge, Chrome e Firefox.

- **Desempenho – O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela inicial – exibição de ofertas e tela de monitoramento de cupons).**

  Esse RNF foi escolhido com o objetivo de garantir uma boa performance na aplicação e poder determinar se o desempenho desses requisitos funcional, principais e mais dispendiosos, serão atendidos.

  Os critérios de aceite são:

  - O sistema deve responder no caso mais dispendioso (a busca da totalidade dos registros na base de dados local) em menos de 2 segundos.

- **Disponibilidade – O sistema deve operar em qualquer período do dia e da noite.**

Esse RNF foi escolhido com o objetivo de assegurar a tolerantes a falhas (failover) nos componentes arquiteturais.

Os critérios de aceite são:

- Os componentes arquiteturais devem ser implantados e executados em clusters, tendo mais de uma instancia para cada componente no cluster de forma distribuída, permitindo a execução de um comando em qualquer dessas instancias destinadas a execução desse comando.

- **Resiliência (*network*) – O sistema deve permanecer operando em condições adversas de conectividade.**

Esse RNF foi escolhido com o objetivo de assegurar o funcionamento do sistema em condições adversas de conectividade.

Os critérios de aceite são:

- O sistema deve permitir a execução de todas as ações do sistema (*commands* e *queries*) tanto *online* como *offline*.
- A sincronização dos *commands* e *queries* do banco de dados local com o banco de dados remoto do usuário deve acontecer com o restabelecimento da conectividade.

- **Escalabilidade manual ou automática**

Esse RNF foi escolhido com o objetivo de assegurar a capacidade dos componentes arquiteturais escalarem vertical e horizontalmente.

Os critérios de aceitação são:

- O sistema deve permitir escalar todos os componentes arquiteturais dentro de seus clusters vertical e horizontalmente.

- A elasticidade dos componentes no cluster podem ocorrer de forma manual ou automática.

5.2. Interfaces/ APIs

A API disponível para que cada usuário possa utilizar para integração com qualquer outro sistema/ferramenta é a API CouchDB do banco de dados remoto do usuário. Essa é uma API muito poderosa que contempla vários recursos, como o *replication*, porém a rota da API mais expressiva é que manipula os documentos **/db/{docid}**, onde o **/db** é userdb-{hex encoded usermail} e **/{docid}** é o id do documento no CouchDB. Assim, para o usermail test@test.com ficará **/userdb-7465737440746573742e636f6d/{docid}**.

Os comandos são criados dentro de uma estrutura padrão de objeto, onde:

```
{
    "service": "Offers",
    "commandJSON": {command object},
    "commandName": "Commands.CreateOffer",
    "type": "command"
}
```

- Service: Nome do serviço responsável pelo comando.
- CommandJSON: Objeto responsável pelos dados do comando.
- CommandName: Representa o tipo do comando.
- Type: Existem dois tipos (commnad e query), os comandos são responsáveis por díspar ações no back-end, já os objetos do tipo query são objetos de consulta apenas.

Segue um exemplo de um comando para criação de oferta.

```
Request:
    PUT /recipes/SpaghettiWithMeatballs HTTP/1.1
    Accept: application/json
    Content-Length: 196
    Content-Type: application/json
    Host: localhost:5984
    {
        "Service": "Offers",
        "CommandJSON": {
                                \"_id\":\"{documentId}\",
                                \"Name\":\"Offer1\",
                                \"EffectiveStartDate\":\"{datetime}\",
                                \"EffectiveEndDate\":\"{datetime}\",
                                \"ListProduct\":[{
                                    \"Guid\":null,
                                    \"Name\":\"Product1\",
                                    \"Image\":\"imageUrl\",
                                    \"Value\":12.0,
                                    \"IsCoupon\":true,
                                    \"AmountCoupon\":10,}]
                            },
        "CommandName": "Commands.CreateOffer",
        "Type": "command"
    }
```

```
Response:
    HTTP/1.1 201 Created
    Cache-Control: must-revalidate
    Content-Length: {length of document}
    Content-Type: application/json
    Date: {datetime of inserted document}
    ETag: {revison}
    Location: http://localhost:5984/recipes/SpaghettiWithMeatballs
    Server: CouchDB (Erlang/OTP)

    {
        "id": "SpaghettiWithMeatballs",
        "ok": true,
        "rev": {revison}
    }
```

Os comandos aceitos são:

- Service: Offers
- CommandName: Commands.CreateOffer
- Type: command

CreateOffer		
Campo	**Tipo**	**Descrição**
_id	*string*	Id CouchDB
Name	*string*	Nome da oferta
EffectiveStartDate	*datetime*	Data inicio da oferta
EffectiveEndDate	*datetime*	Data fim da oferta
ListProduct	*list<product>*	Lista de produtos.

Product		
Campo	Tipo	Descrição
Guid	*string*	Id do produto
Name	*string*	Nome do produto
Image	*string*	Url da imagem do produto
Value	*decimal*	Valor do produto
IsCoupon	*bool*	Se o produto é oferecido na modalidade cupom.
AmountCoupon	*int*	Quantidade de cupom.

- Service: Offers
- CommandName: Commands.RequestCoupon
- Type: command

RequestCoupon		
Campo	**Tipo**	**Descrição**
IdOffer	*string*	Id da oferta
GuidProduct	*string*	Identificador do produto
UserEmail	*string*	E-mail do usuário

- Service: Offers
- CommandName: Commands.Delivered
- Type: command

Delivered		
Campo	**Tipo**	**Descrição**
IdOffer	*string*	Id da oferta
GuidProduct	*string*	Identificador do produto
UserEmail	*string*	E-mail do usuário

- Service: Offers
- CommandName: Commands.FinishedStock
- Type: command

FinishedStock		
Campo	**Tipo**	**Descrição**
IdOffer	*string*	Id da oferta
GuidProduct	*string*	Identificador do produto
UserEmail	*string*	E-mail do usuário

- Service: Offers
- CommandName: Commands.InStock
- Type: command

InStock		
Campo	**Tipo**	**Descrição**
IdOffer	*string*	Id da oferta
GuidProduct	*string*	Identificador do produto
UserEmail	*string*	E-mail do usuário

- Service: PyGoogleImg
- CommandName: Commands.GoogleSearch
- Type: command

GoogleSearch		
Campo	**Tipo**	**Descrição**
Search	*string*	Nome do produto a ser localizado

As queries são:

O CouchDB disponibiliza a API **/db/_find** para facilitar as buscas de documentos.

Offer		
Campo	**Tipo**	**Descrição**
_id	*string*	Id CouchDB
Name	*string*	Nome da oferta
Location	*string*	Localização do comércio
EffectiveStartDate	*datetime*	Data inicio da oferta
EffectiveEndDate	*datetime*	Data fim da oferta
Type	*string*	Type: “Offer”
CqrsType	*string*	Type: “query”
ListProduct	*list<product>*	Lista de produtos.

Product		
Campo	Tipo	Descrição
Guid	*string*	Id do produto
Name	*string*	Nome do produto
Image	*string*	Url da imagem do produto
Value	*decimal*	Valor do produto
IsCoupon	*bool*	Se o produto é oferecido na modalidade cupom.
AmountCoupon	*int*	Quantidade de cupom.

GoogleSearchImg		
Campo	Tipo	Descrição
Guid	*string*	Id do produto: GoogleSearchImg
Type	*string*	Type: “GoogleSearchImg”
CqrsType	*string*	Type: “query”
Search	*string*	Nome do produto buscado
ListUrlImages	*List<string>*	Lista de cinco urls do ultimo produto buscado.

6. Avaliação da Arquitetura

6.1. Análise das abordagens arquiteturais

As principais características dessa proposta arquitetural é conceder ao sistema a capacidade de executar todas as suas funções em condições adversas de conectividade sem prejuízo ao usuário, com desempenho de execução adequados dessas atividades, mesmo em momentos de pico ou aumento substancial da quantidade de usuários simultâneos. Esta proposta também prioriza a alta disponibilidade do sistema, pois suas funções de negócio envolvem transações comercias.

6.2. Identificação dos atributos de qualidade

Os atributos identificados estão relacionados aos requisitos listados na seção anterior: Segurança, acessibilidade, desempenho, disponibilidade, resiliência e escalabilidade.

6.3. Cenários

Cenário 1: Ao realizar o acesso a uma URL ou página, o sistema deve apresentar altos padrões de segurança necessário, garantindo que o usuário possa acessar apenas os dados presentes na sua base de dados remota ou local. Garantindo assim a segurança e confidencialidade das informações estando em de acordo com um dos requisitos não funcionais.

Cenário 2: Ao realizar o acesso a aplicação através de um dispositivo móvel ou desktop com resolução reduzida, utilizando os browser principais como IE, Chrome e Firefox, a tela do usuário deverá se adaptar automaticamente, redimensionando seus links, botões e tabela de dados de acordo com a resolução, estando de acordo com acessibilidade necessária para atender os requisitos não funcionais de acessibilidade.

Cenário 3: O sistema deve ser rápido em suas funcionalidades principais, que são também as mais dispendiosas (tela inicial – listagem de ofertas e tela de monitoramento de cupons – listagem de cupons). O sistema deve responder no caso mais dispendioso (a busca da totalidade dos registros na base de dados local) em menos de 2 segundos.

Cenário 4: O sistema deve operar em qualquer período do dia ou da noite. Dessa forma, os componentes arquiteturais devem ser implantados e executados em clusters, tendo

mais de uma instancia para cada componente em um cluster de forma distribuída, assegurando a tolerância a falhas e alta disponibilidade.

Cenário 5: O sistema deve permanecer operando em condições adversas de conectividade. O usuário logado na aplicação deve continuar utilizando o sistema sem perceber que houve uma indisponibilidade da rede de dados. Todas as solicitações dos usuários relativas às consultas no sistema (*querys*) devem ser atendidas normalmente, não havendo impacto nessas funcionalidades centrais. As solicitações de comandos (*commands*) devem ser enfileiradas no banco de dados local do usuário para posterior sincronização.

Cenário 6: O sistema deve permanecer respondendo normalmente, mesmo com o crescimento da utilização de recursos do sistema, seja por aumento da quantidade média de usuários simultâneos ou por picos de utilização do sistema. O sistema devera estar preparado para ser escalável vertical e horizontalmente no cluster de *microservices* de forma manual ou automática. As instâncias de banco de dados devem ser escaladas de forma crescente apenas, manual ou automaticamente, de acordo com os indicadores de desempenho (*key performance indicator*) configurados.

Na priorização foi utilizado o método de árvore de utilidade reduzida e com prioridades. Foi categorizado de acordo os atributos de qualidade a que estão relacionados e então classificados em função de sua importância e complexidade, considerando a percepção de negócio e arquitetura. As duas variáveis de priorização "Importância" e "Complexidade", apresentadas nas colunas "IMP." e "COM." respectivamente foram classificadas em alta (A), média (M) e baixa (B) de acordo com as características do requisito.

Atributos de Qualidade	Cenários	IMP.	COM.
Segurança	(*Database per user*) – O sistema deve apresentar altos padrões de segurança.	A	M
Acessibilidade	O sistema deve suportar ambientes Web responsivos e ambientes móveis.	M	B
Desempenho	O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela inicial – exibição de	M	A

	ofertas e tela monitoramento de cupons).		
Disponibilidade	O sistema deve operar em qualquer período do dia e da noite.	A	A
Resiliência	(*network*) – O sistema deve permanecer operando em condições adversas de conectividade.	M	A
Escalabilidade	O sistema de escalar vertical e/ou horizontalmente no cluster de forma manual ou automática.	A	A

6.4. Avaliação

São analisados nesse processo de avaliação os cenários identificados no item 5.1. O objetivo é determinar os riscos, não riscos, pontos de sensibilidade e trade-offs e as evidências mostrando o requisito de qualidade sendo atendido.

- **Cenário 1**

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	(*database per user*) – O sistema deve apresentar altos padrões de segurança
Preocupação:	
Impossibilitar o acesso à base de dados remota do usuário que não esteja autenticado ou que autenticado em outra base de dados remota do usuário. Garantindo assim, o acesso/sincronização da base de dados local do usuário apenas com sua base de dados remota.	
Cenário(s):	
Cenário 1	
Ambiente:	
Usuário utilizando o sistema com carga normal.	
Estimulo:	
Acessar o sistema sem estar autenticado.	

Mecanismo:	
Autenticação e Autorização (CouchDB Basic Authentication e CouchDB Cookie Authentication).	
Medida de Resposta:	
Não permitir a conexão de um usuário autenticado a uma base de dados de outro usuário e não permitir o acesso à base de dados de usuário quando não autenticado.	
Considerações sobre a arquitetura:	
Riscos:	Falhas no sistema de autenticação e autorização podem permitir o acesso indevido a informações não autorizadas.
Pontos de Sensibilidade:	Manter a opção **require_valid_user** marcado como **true** para bloquear o acesso anônimo e garantir que um usuário valido esteja autenticado para qualquer *request* ao servidor CouchDB (http://docs.couchdb.org/en/2.2.0/config/auth.html#chttpd/require_valid_user). Manter o opção **couch_peruser** marcada como **true** para assegurar que a criação de um novo usuário no CouchDB também irá criar uma base de dados remota privada para esse usuário. O CouchDB garante assim, o acesso desse usuário à apenas essa base de dados (http://docs.couchdb.org/en/2.2.0/config/couch-peruser.html#couch_peruser).
Tradeoff:	Não existe.

- **Evidências do Cenário 1**

  Evidência do CouchDB Fauxton que demonstra como o usuário autenticado pelo CouchDB com as opções **couch_peruser** e **require_valid_user** habilitadas apenas poderão acessar sua base de dados especifica.

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=26

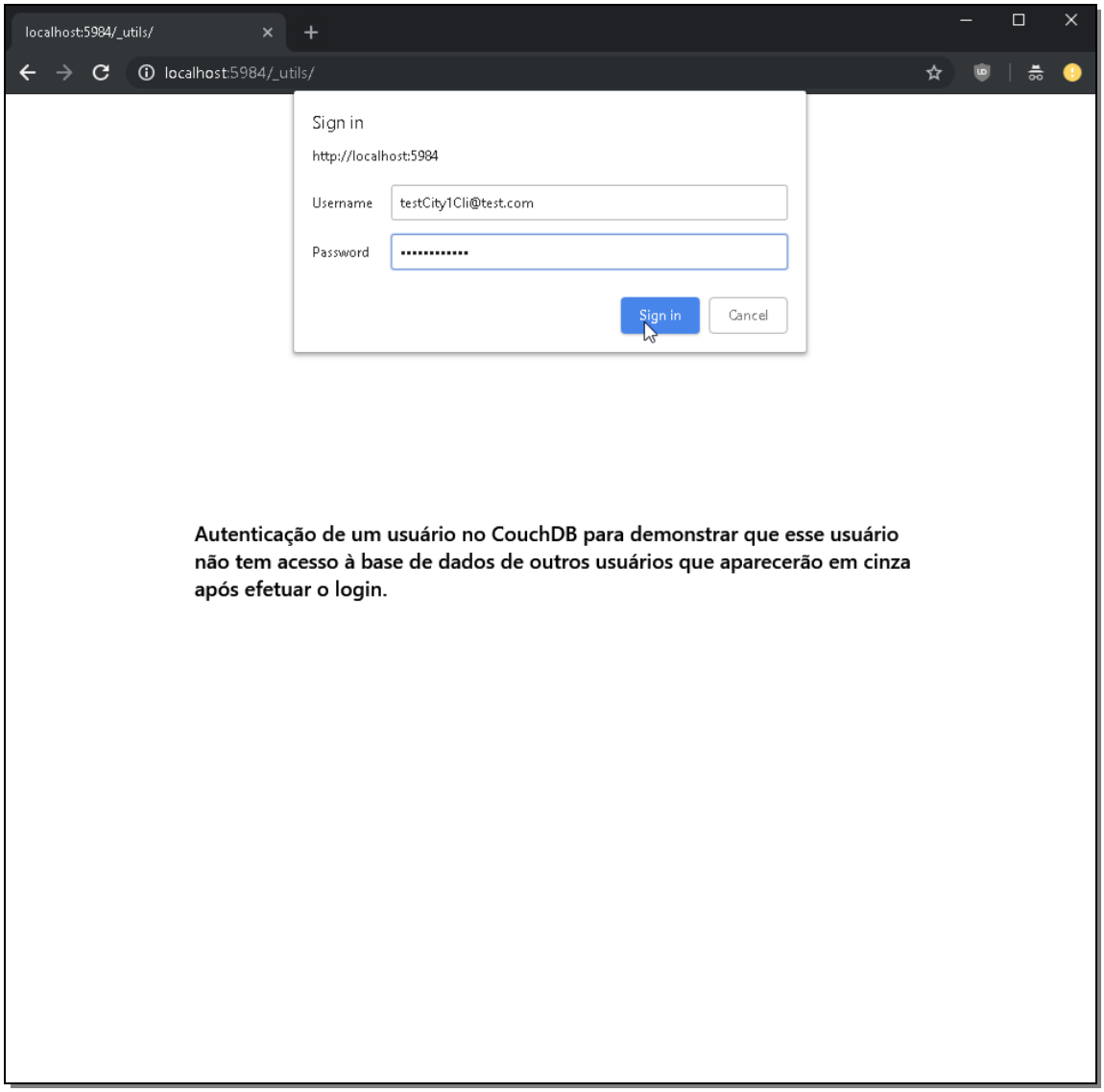


Figura 6 - Login de um usuário no CouchDB Fauxton

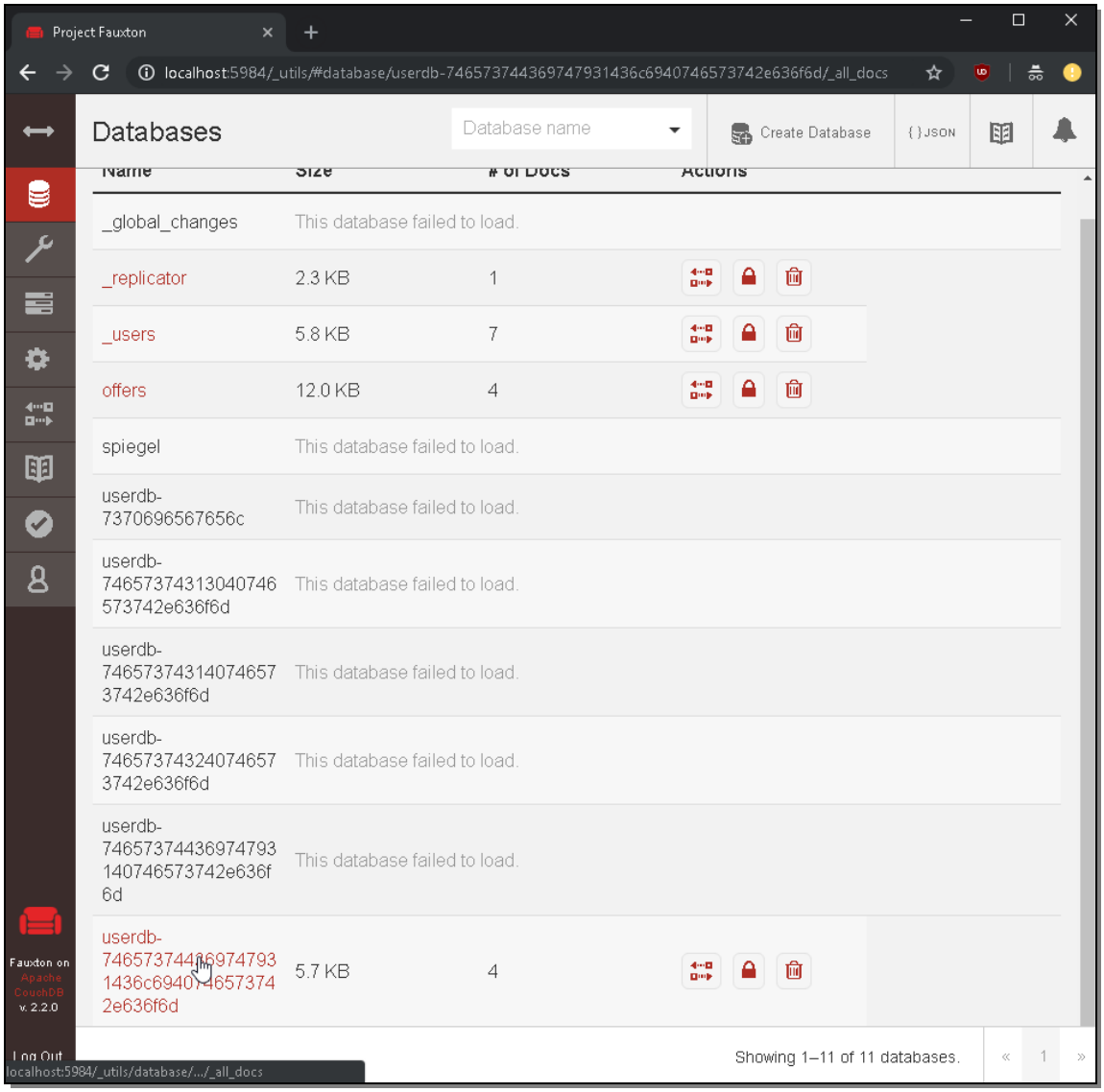


Figura 7 - Usuário logado no CouchDB não tem acesso à base de dados de outros usuários

- **Cenário 2**

Atributo de Qualidade:	Acessibilidade
Requisito de Qualidade:	O sistema deve suportar ambiente web responsivos e ambientes móveis.
Preocupação:	
O sistema deve redimensionar seus objetos de acordo com o tamanho da tela.	
Cenário(s):	
Cenário 2.	
Ambiente:	
Aplicativo cliente em operação normal.	
Estimulo:	
Usuário acessando o sistema pela aplicação cliente (*front-end*).	
Mecanismo:	
Layout responsivo com flexbox e media queries (*Bootstrap*).	
Medida de Resposta:	
O sistema deve se adaptar a resoluções de tela do dispositivo móvel, desktop ou tablet.	
Considerações sobre a arquitetura:	
Riscos:	A ausência do componente *Bootstrap*, na aplicação cliente, por alguma falha em seu carregamento, acarretará falha na aplicação do layout sem prejuízo para os dados do sistema.
Pontos de Sensibilidade:	A disponibilidade de uma hospedagem publica do componente *Bootstrap* podem conflitar com o acordo de nível de serviço (*SLA)* pretendido para a aplicação.
Tradeoff:	A escolha entre uma rede de distribuição de conteúdo *(CDN)* publica ou privada para a disponibilizar os pacotes utilizados no aplicativo vai depender do *SLA* pretendido no aplicativo.

- **Evidências do Cenário 2**

  Evidência do navegador web em *Device Mode* (simulação de dispositivos móveis) mostrando a responsividade e a aplicação se adaptando em ambientes mobile simulados.

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=4

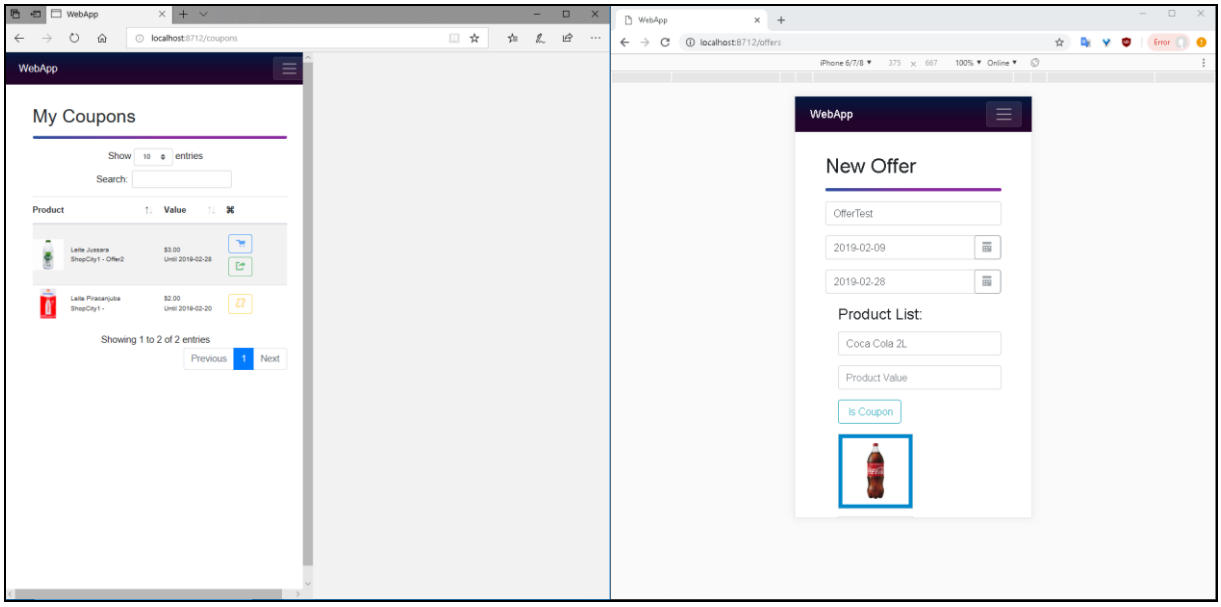


Figura 8 - Navegador Edge e Chrome em Device Mode

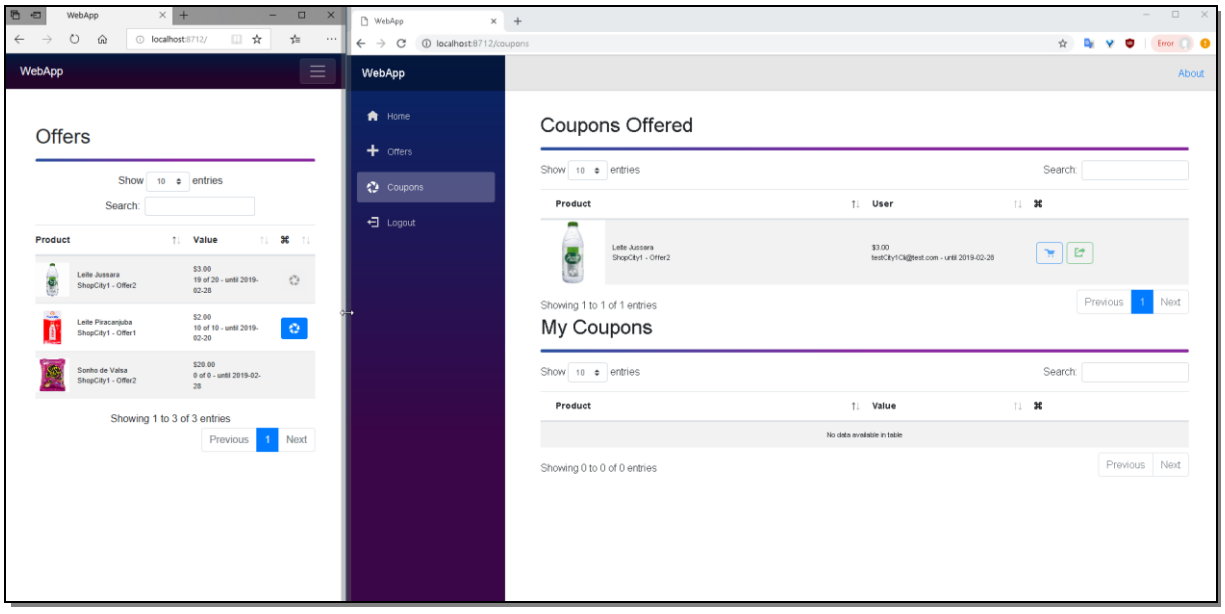


Figura 9 - Testando a responsividade da aplicação Blazor (WebAssembly) no Edge

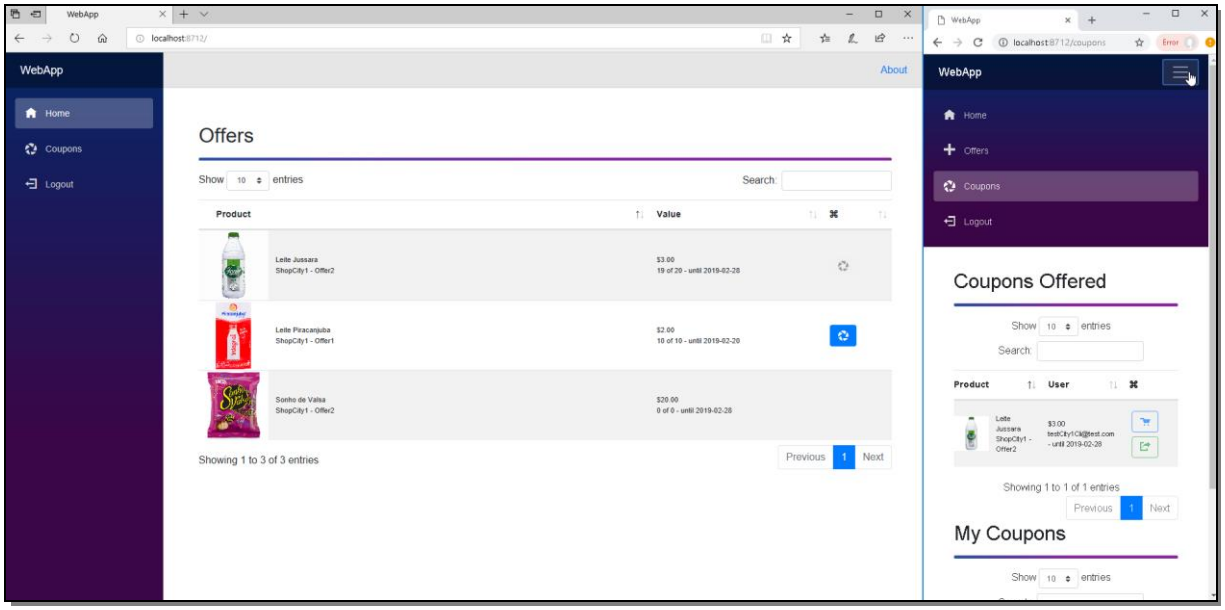


Figura 10 - Testando a responsividade da aplicação Blazor (WebAssembly) no Chrome

- **Cenário 3**

<table>
<tr><td>Atributo de Qualidade:</td><td>Desempenho</td></tr>
<tr><td>Requisito de Qualidade:</td><td>O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela inicial – exibição de ofertas e tela monitoramento de cupons).</td></tr>
<tr><td colspan="2">Preocupação:</td></tr>
<tr><td colspan="2">O sistema deve apresentar desempenho satisfatório dentro dos limites aceitáveis.</td></tr>
<tr><td colspan="2">Cenário(s):</td></tr>
<tr><td colspan="2">Cenário 3.</td></tr>
<tr><td colspan="2">Ambiente:</td></tr>
<tr><td colspan="2">Sistema em operação com carga normal.</td></tr>
<tr><td colspan="2">Estimulo:</td></tr>
<tr><td colspan="2">Usuário acessando ou consultando ofertas na tela inicial ou na tela de monitoramento de cupons do sistema (as tarefas mais custosas do sistema).</td></tr>
<tr><td colspan="2">Mecanismo:</td></tr>
<tr><td colspan="2">NoBackend, CouchDB/PouchDB Sync Protocol e CQRS</td></tr>
<tr><td colspan="2">Medida de Resposta:</td></tr>
<tr><td colspan="2">O sistema deve responder no caso mais dispendioso (leitura local da totalidade dos registros) em menos de 2 segundos.</td></tr>
<tr><td colspan="2">Considerações sobre a arquitetura:</td></tr>
<tr><td>Riscos:</td><td>PouchDB é totalmente testado e suportado em modernos Browsers:
Firefox 29+ (Including Firefox OS and Firefox for Android)
Chrome 30+
Safari 5+
Internet Explorer 10+
Opera 21+
Android 4.0+
iOS 7.1+
Windows Phone 8+</td></tr>
</table>

	• PouchDB também roda em Cordova/PhoneGap, NW.js, Electron e Chrome apps. Para suporte a *Browsers* antigos veja: https://pouchdb.com/learn.html#browser_support
Pontos de Sensibilidade:	Uma configuração que pode aumentar sensivelmente no tempo de busca dos registros, porém ainda dentro do limite aceitável de 2 segundos é a **PouchDB auto_compaction** marcada com **false**, essa configuração marcada como **true** faz a versão local da base de dados do usuário mantenha apenas a ultima versão de cada documento diminuindo sensivelmente o tempo de busca e o tamanho da base de dados local do usuário (https://pouchdb.com/guides/compact-and-destroy.html#auto-compaction).
Tradeoff:	Não existe.

- **Evidências do Cenário 3**

  Evidência do depurador web com o tempo total da leitura e obtenção de todos os documentos da base de dados local PouchDB, representado as operações de leitura da aplicação sendo está a mais custosa da aplicação, pois obtém a totalidade dos registros da base de dados.

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=52

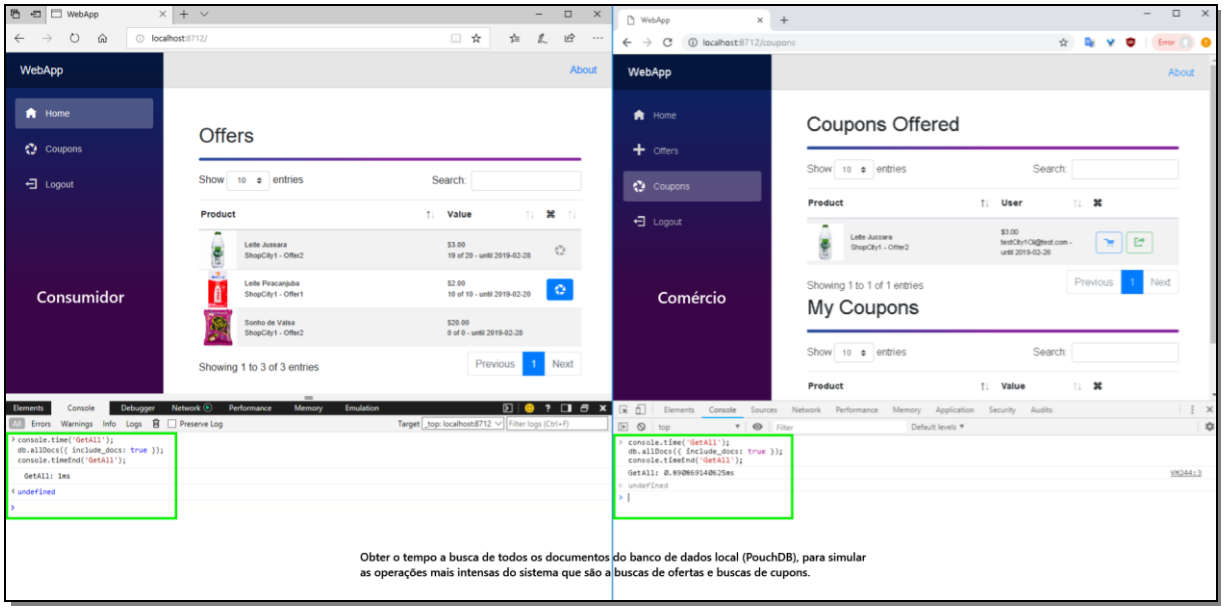


Figura 11 - Performance Client-Side, buscando todos os documentos no PouchDB, nos perfis Consumidor e Comerciante

- **Cenário 4**

Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema deve operar em qualquer período do dia e da noite.
Preocupação:	
A indisponibilidade de algum *node* do cluster de banco de dados ou *node* do cluster de componentes arquiteturais/negócio (*microservices*) não devem impactar na utilização do sistema, em qualquer de suas funcionalidades.	
Cenário(s):	
Cenário 4.	
Ambiente:	
Sistema em operação com carga normal e manutenções ou atualizações sendo aplicadas aos *nodes* ou componentes do cluster de banco de dados de componentes arquiteturais/negócio (*microservices*).	
Estimulo:	
Diversos usuários estão utilizando o sistema com manutenção em alguns *node* do cluster ou atualização de algum(s) componente(s) arquitetural ou de negócio.	
Mecanismo:	
Cluster CouchDB, Cluster Spiegel e Cluster Service Fabric.	
Medida de Resposta:	
Todos os usuários autenticados na aplicação devem continuar utilizando o sistema sem perceber que houve uma queda de um dos *nodes* do servidor de aplicação.	
Considerações sobre a arquitetura:	
Riscos:	Os clusters devem ter no mínimo 3 *nodes* para oferecer o nível de confiabilidade mais baixo, o bronze. Clusters sem o nível mínimo de confiabilidade tornam o processo de atualização e manutenção fortemente propício a erros. É desejável o nível de confiabilidade Gold (7 ou 8 nodes) ou Platinum (acima de 9 nodes) - https://docs.microsoft.com/pt-br/azure/service-fabric/service-fabric-cluster-capacity#recommendations-for-the-reliability-tier.
Pontos de Sensibilidade:	A distribuição das instancias de serviços (*microservices*) e banco de dados no cluster devem ser configuradas para assegurar que haja no mínimo três instancias de um serviço ou

	banco de dados distribuídos em *nodes* diferentes. O arquivo ApplicationManifest.xml é o responsável por orientar o orquestrador do Service Fabric no que diz respeito a distribuição das instancias dos *microservices* no cluster. E o arquivo etc/local.ini do CouchDB é responsável pelo gerenciamento de *shards* e *replicas* dos bancos de dados no cluster do CouchDB.
Tradeoff:	A decisão da quantidade de instancias ou replicas dos bancos de dados que um node é capaz de atender vai depender da capacidade do node e do consumo de recursos que um serviço consome. Essa distribuição também pode ser configurada para que aconteça de forma automática por meio de indicadores coletados nos nodes, como por exemplo, o percentual de uso de cpu e memória física.

- **Evidências do Cenário 4**

  Evidência do Service Fabric Explorer mostrando a quantidade de node no cluster. A disponibilidade dos clusters CouchDB e Spiegel funcionam de forma semelhante ao do cluster Service Fabric mostrado aqui.

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=198

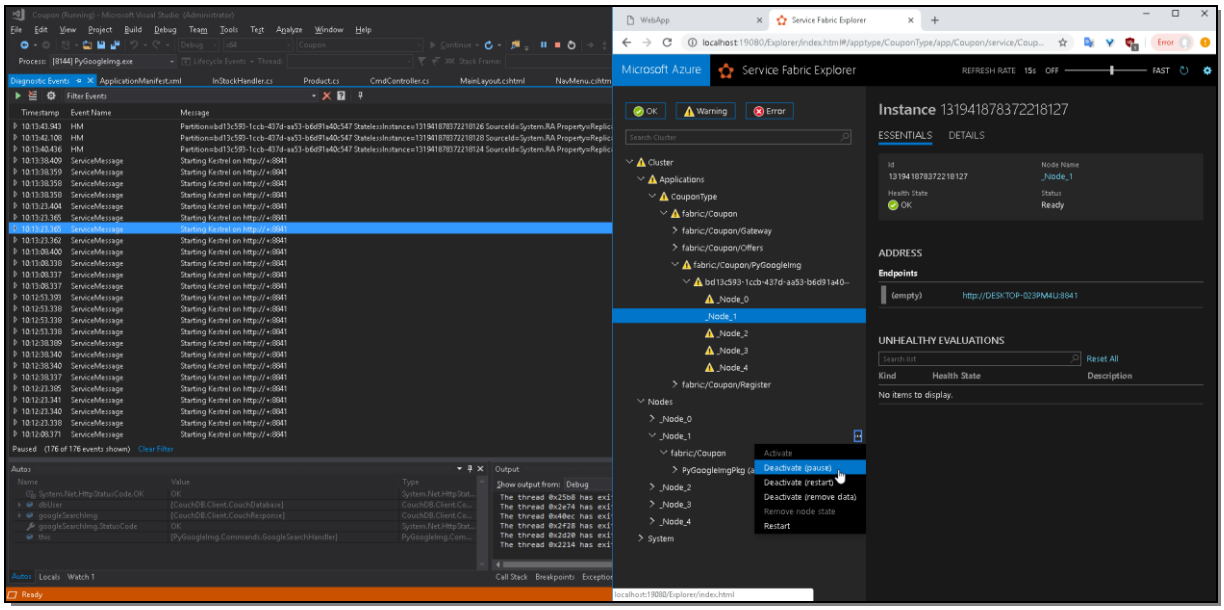


Figura 12 - Procedimento de desativação do node1 do cluster Service Fabric composto por 5 nodes

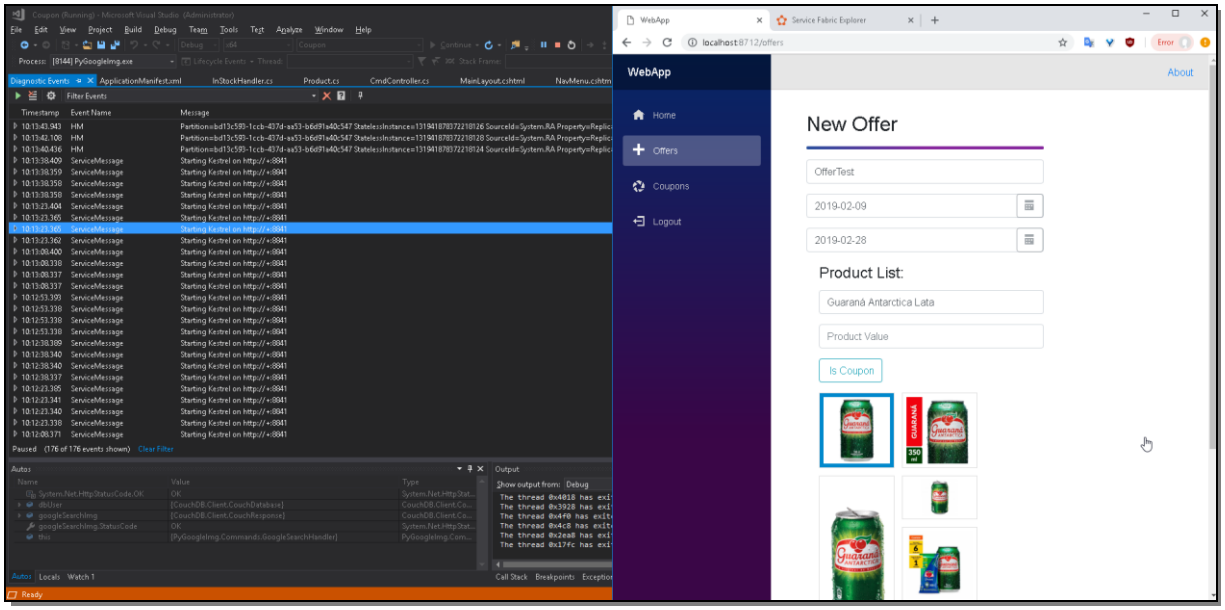


Figura 13 - Busca de imagens do serviço PyGoogleImg com o node1 desativado

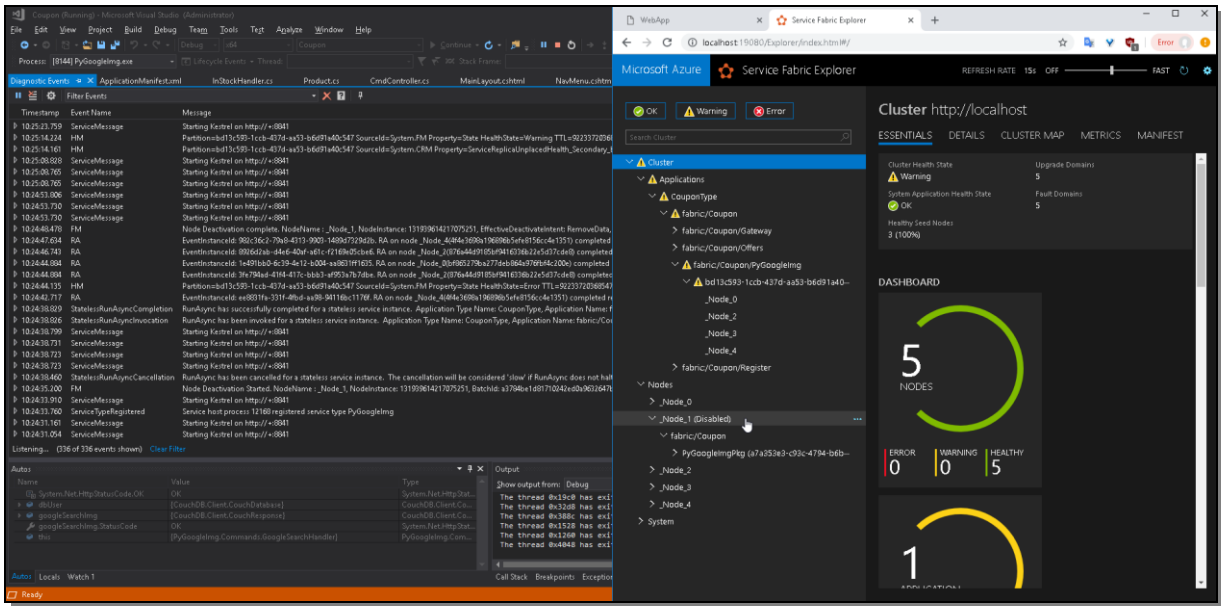


Figura 14 - Service Fabric Explorer exibindo o node1 desativado

- **Cenário 5**

Atributo de Qualidade:	Resiliência
Requisito de Qualidade:	(*network*) – O sistema deve permanecer operando em condições adversas de conectividade.
Preocupação:	
As conexões de rede dos dispositivos móveis são totalmente instáveis devido à mobilidade. Logo uma transação que iniciou em uma conexão de rede 3G, por exemplo, pode terminar em uma conexão wi-fi. Essas alternâncias de conexões ou a ausência de uma não devem impactar na utilização do sistema, em qualquer de suas funcionalidades.	
Cenário(s):	
Cenário 5.	
Ambiente:	
Sistema em operação com carga normal.	
Estimulo:	
Instabilidade na conectividade da aplicação cliente com o servidor.	
Mecanismo:	
PouchDB, CouchDB, Couch Sync Protocol, CQRS e Event Sourcing.	
Medida de Resposta:	
O usuário da aplicação deve continuar utilizando o sistema sem perceber que houve uma indisponibilidade da rede de dados. Todas as solicitações dos usuários relativas às consultas no sistema (*querys*) devem ser atendidas normalmente, não havendo impacto nessas funcionalidades centrais. As solicitações de comandos (*commands*) devem ser enfileiradas no banco de dados local do usuário para posterior sincronização.	
Considerações sobre a arquitetura:	
Riscos:	Sem o uso do estilo arquitetural CQRS (*command and query responsibility segregation*) em conjunto com o padrão Event Sourcing, podem ocorrer conflitos no processo de sincronização, uma vez que um documento pode ser alterado pelo cliente e pelo servidor simultaneamente antes do processo de sincronização, o que ocasionará conflito nos documentos.
Pontos de Sensibilidade:	A não utilização do CQRS com Event Sourcing pode gerar conflitos no processo de sincronização entre a base de dados local e remota do usuário.

Tradeoff:	A decisão por outro estilo arquitetural e/ou padrão devem sempre levar em consideração as resoluções de conflitos previstas pelo CouchDB e PouchDB.

- **Evidências do Cenário 5**

  Evidência da aplicação funcionando normalmente com o navegador web em *Offline Mode* (desconectado da rede, ou seja, sem trafego de dados).

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=72

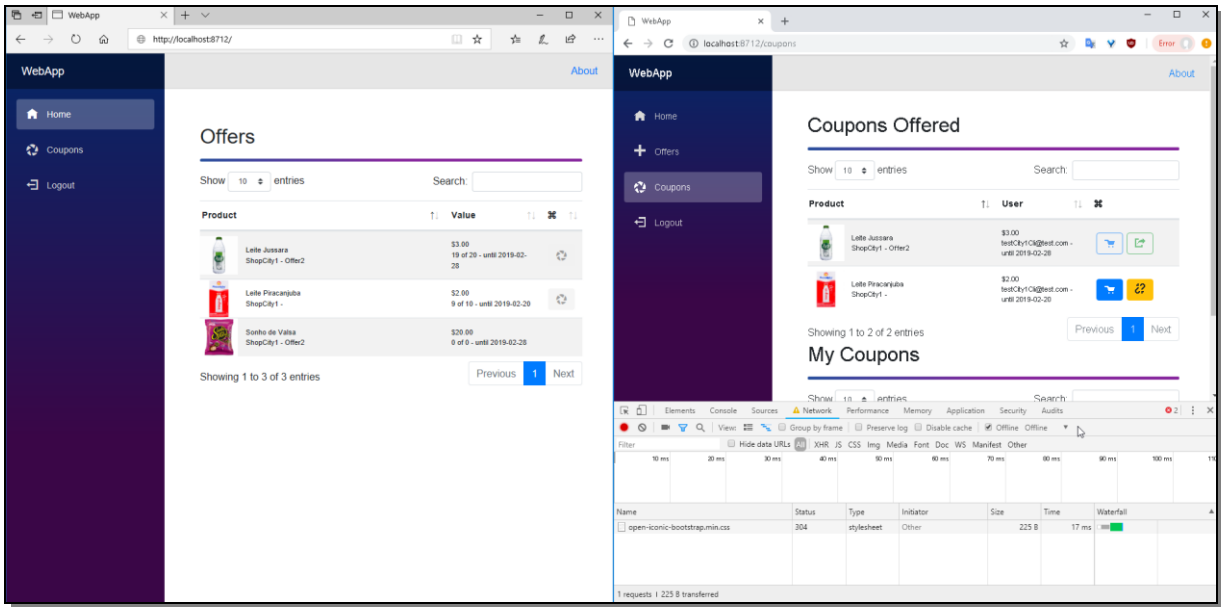


Figura 15 - Aplicação do perfil Comerciante em offline mode

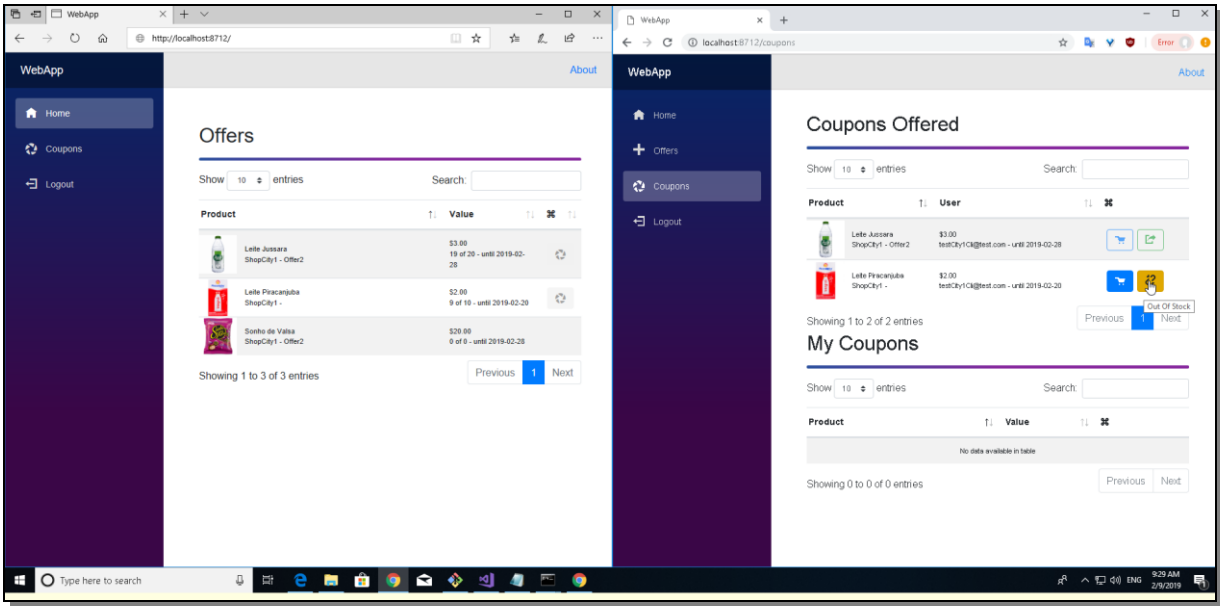


Figura 16 - Comando de produdo fora do estoque sendo disparado em modo offline para simular a peda de conexão do cliente

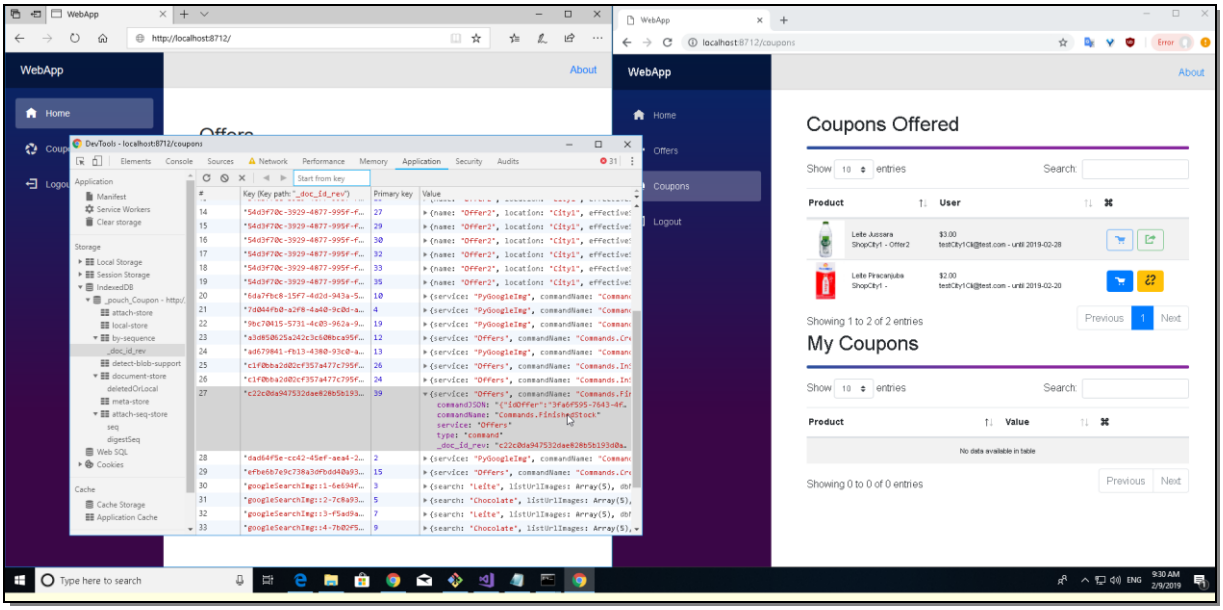


Figura 17 – Utilização do sistema normalmente, com comando inserido na base de dados local (PouchDB) do cliente com sucesso, aguardando o retorno da conexão para sincronização com sua base de dados remota (CouchDB)

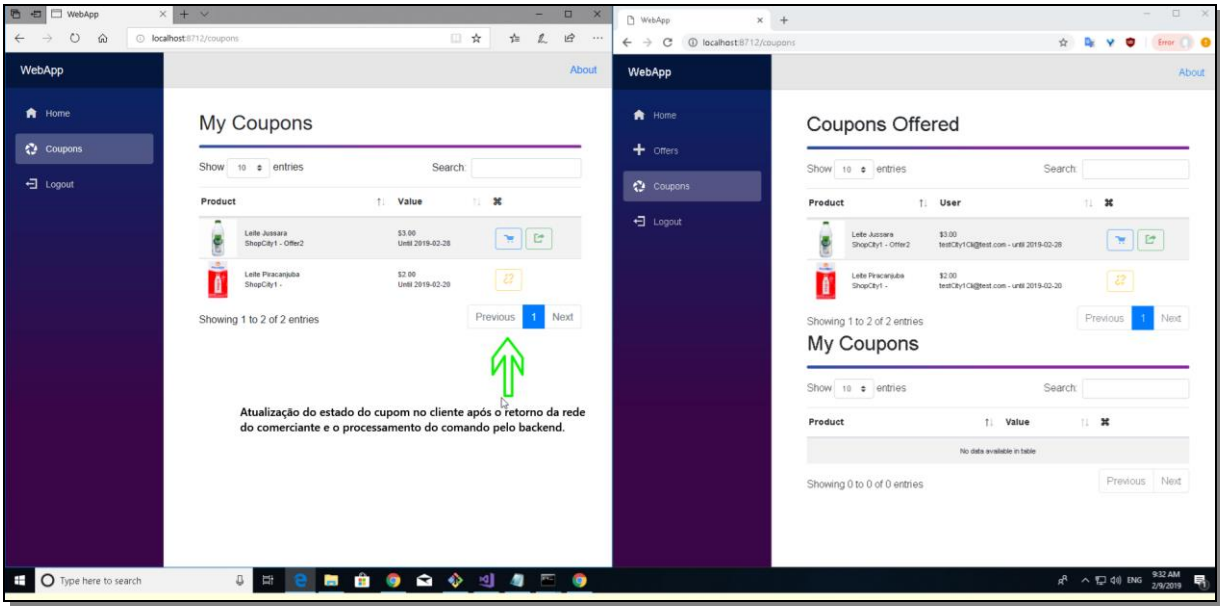


Figura 18 - Ao retorno da conexão o comando de "fora do estoque" é processado pelo backend e refletido na aplicação com perfil Consumidor

- **Cenário 6**

Atributo de Qualidade:	Escalabilidade
Requisito de Qualidade:	O sistema deve ser capaz de escalar vertical e horizontalmente no cluster, de forma manual ou automaticamente.
Preocupação:	
O sistema deve suportar o aumento gradual ou momentos de pico de utilização simultânea do sistema sem que haja alteração de código fonte para isso.	
Cenário(s):	
Cenário 6.	
Ambiente:	
Usuário utilizando o sistema com carga média acima da normal ou em momentos de pico, ocasionado por um aumento rápido da quantidade de usuários simultâneos no sistema.	
Estimulo:	
Crescimento da utilização de recursos do sistema por aumento da quantidade média de usuários simultâneos ou por picos de utilização do sistema.	
Mecanismo:	
Cluster CouchDB, Cluster Spiegel e Cluster Service Fabric.	
Medida de Resposta:	
O sistema deverá ser capaz de fazer escalonamento vertical e horizontal no *cluster* de *microservices* por meio do orquestrador de *microservices* e do cluster do banco de dados por meio do sistema gerenciado de banco de dados distribuído. As instâncias de banco de dados devem ser escaladas de forma crescente apenas, de acordo com os indicadores de desempenho (*key performance indicator*) configurados.	
Considerações sobre a arquitetura:	
Riscos:	Os clusters devem ter no mínimo 3 *nodes* para oferecer o nível de confiabilidade mais baixo, o bronze. Clusters sem o nível mínimo de confiabilidade tornam o processo de atualização e manutenção fortemente propício a erros. É desejável o nível de confiabilidade Gold (7 ou 8 nodes) ou Platinum (acima de 9 nodes) - https://docs.microsoft.com/pt-br/azure/service-fabric/service-fabric-cluster-capacity#recommendations-for-

	the-reliability-tier.
Pontos de Sensibilidade:	A distribuição das instancias de serviços (*microservices*) e banco de dados no cluster devem ser configuradas para assegurar que haja no mínimo três instancias de um serviço ou banco de dados distribuídos em *nodes* diferentes. O arquivo ApplicationManifest.xml é o responsável por orientar o orquestrador do Service Fabric no que diz respeito a distribuição das instancias dos *microservices* no cluster. E o arquivo etc/local.ini do CouchDB é responsável pelo gerenciamento de *shards* e *replicas* dos bancos de dados no cluster do CouchDB.
Tradeoff:	A decisão da quantidade de instancias ou replicas dos bancos de dados que um node é capaz de atender vai depender da capacidade do node e do consumo de recursos que um serviço consome. Essa distribuição também pode ser configurada para que aconteça de forma automática por meio de indicadores coletados nos nodes, como por exemplo, o percentual de uso de cpu e memória física.

- **Evidências do Cenário 6**

  Evidência do Cluster Service Fabric com um e mais nodes. A escalabilidade horizontal/vertical dos clusters CouchDB e Spiegel funcionam de forma semelhante ao do cluster Service Fabric mostrado aqui. Também é possível configurar a escalabilidade automática/elástica nestes clusters.

  Em vídeo: https://youtu.be/E2hZHNWuFO8?t=152

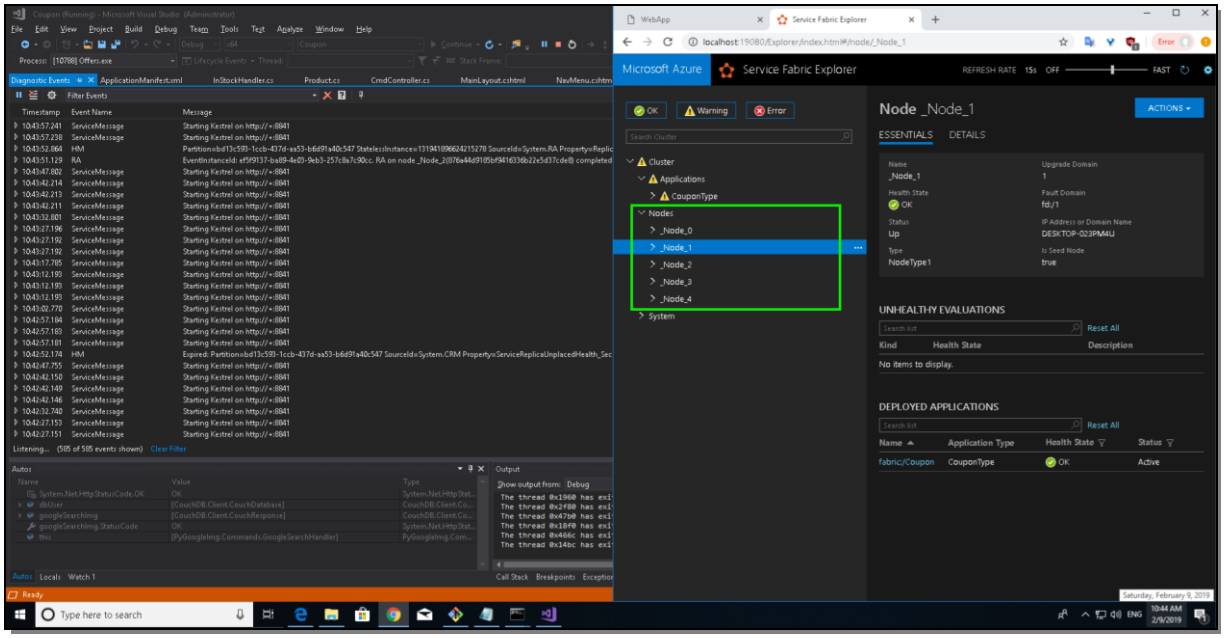


Figura 19 - Service Fabric cluster com 5 nodes

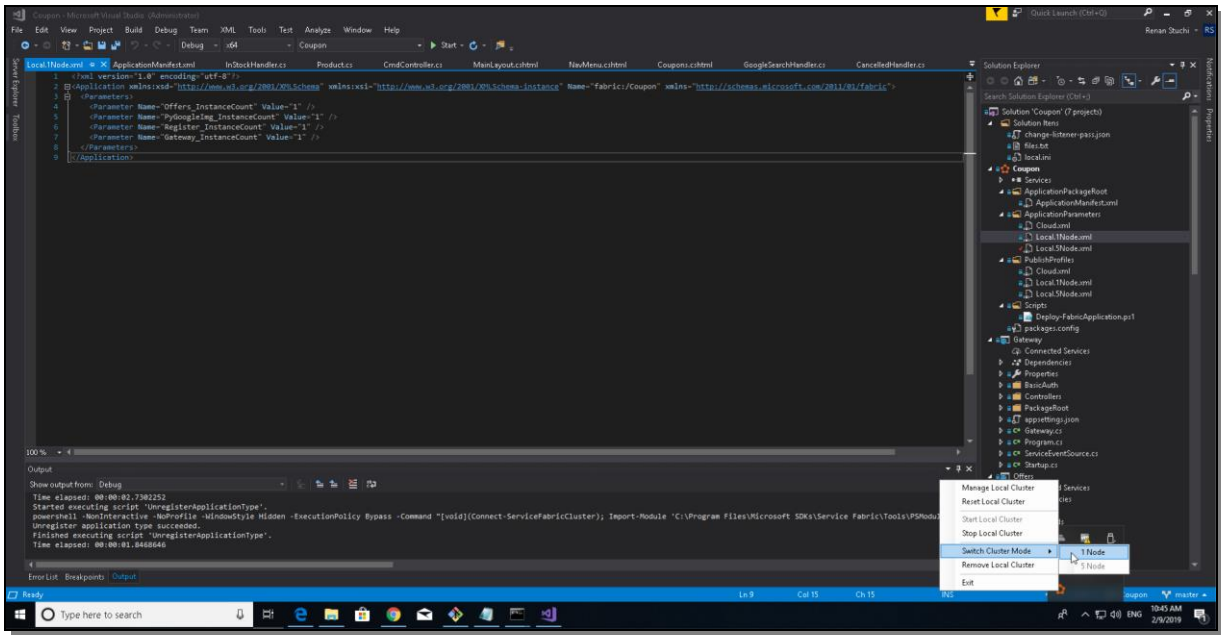


Figura 20 - Configurando Service Fabric cluster para 1 node

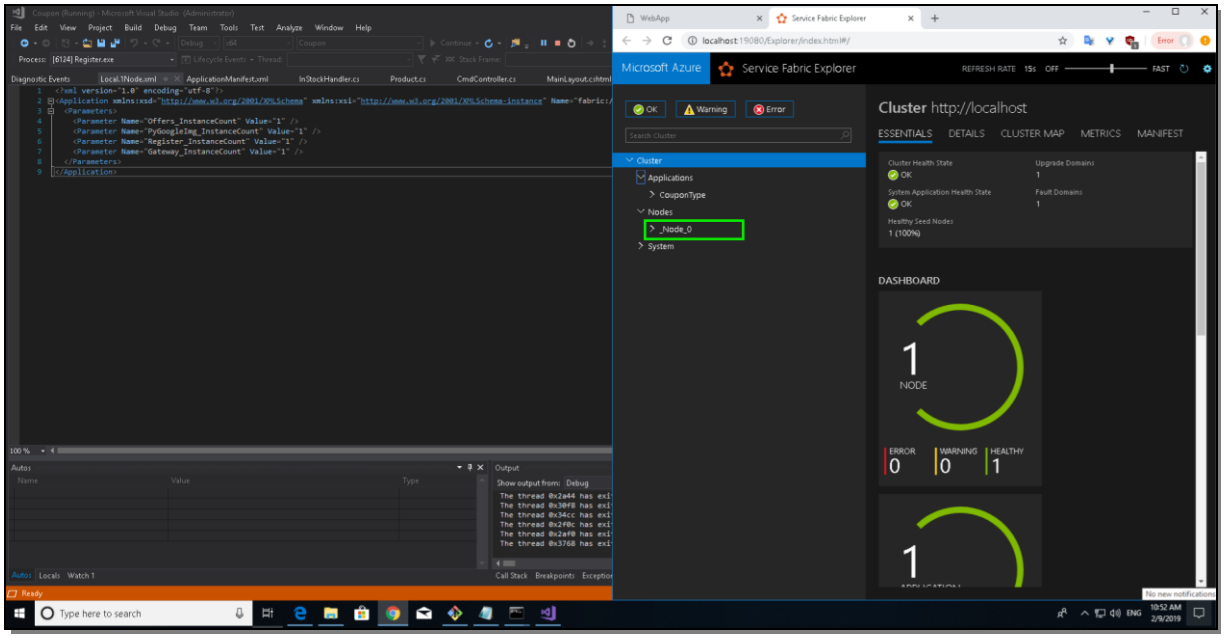


Figura 21 - Service Fabric cluster com 1 node

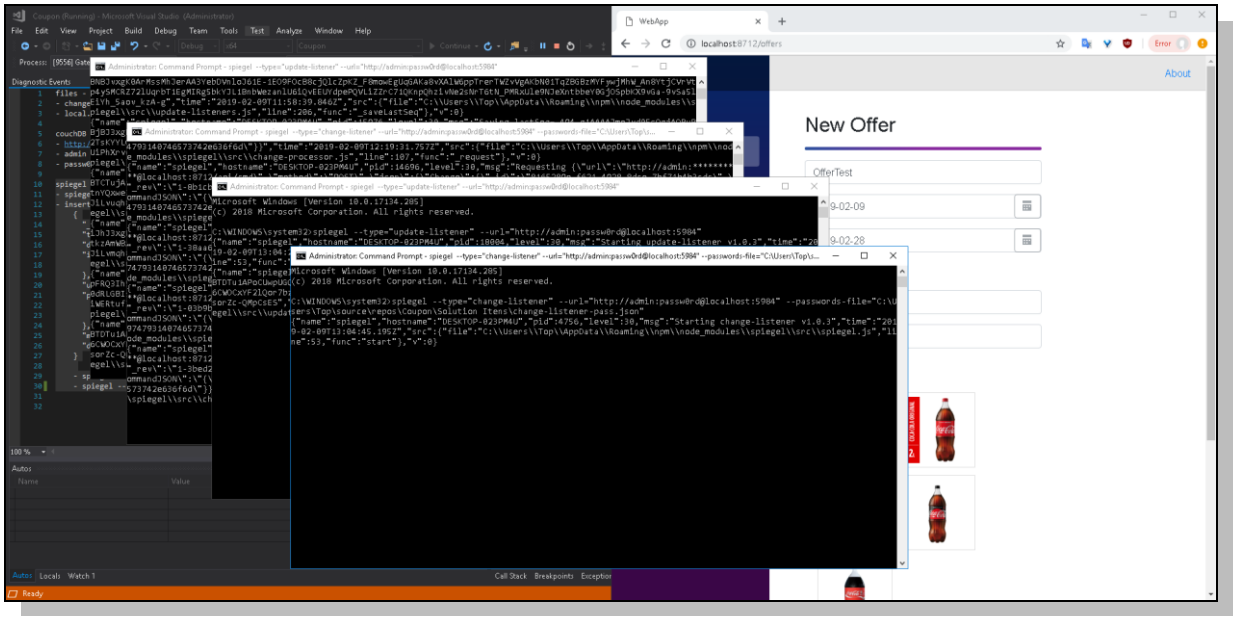


Figura 22 - Spiegel com duas instâncias do "update-listener" e 2 do "change-listener"

6.5. Resultado

Considerando os atributos de qualidade, o objetivo da validação arquitetural foi analisar esses atributos. Verifiquei que a arquitetura proposta atende as necessidades do projeto com possíveis melhorias. Avaliação permitiu que fosse possível concretizar de forma

mais objetiva os testes e cenários para definir pontos fortes e pontos fracos nessa avaliação. Nessa avaliação considerei os seguintes requisitos de qualidades no quadro abaixo.

Atributos de Qualidade	Requisitos Não Funcionais	Testado	Homologado
Segurança	RNF1: (*Database per user*) – O sistema deve apresentar altos padrões de segurança.	SIM	SIM
Acessibilidade	RNF2: O sistema deve suportar ambientes Web responsivos e ambientes móveis.	SIM	SIM
Desempenho	RNF3: O sistema deve ser rápido em suas funcionalidades principais, que são também as mais laboriosas (tela inicial – exibição de ofertas e tela monitoramento de cupons).	SIM	SIM
Disponibilidade	RNF4: O sistema deve operar em qualquer período do dia e da noite.	SIM	SIM
Resiliência	RNF5: (*network*) – O sistema deve permanecer operando em condições adversas de conectividade.	SIM	SIM
Escalabilidade	RNF6: O sistema de escalar vertical e/ou horizontalmente no cluster de forma manual ou automática.	SIM	SIM

Avaliando a arquitetura proposta para esse projeto, foi possível identificar alguns pontos importantes. Com a implementação das três camadas do sistema em cluster (banco de dados distribuído – cluster CouchDB, *message oriented middleware* – cluster Spiegel e microsserviços da aplicação – cluster Service Fabric) facilita muito o atendimento à três requisitos não funcionais (Desempenho, Disponibilidade e Escalabilidade). Uma melhoria no *middleware* de mensageria é a utilização de um *dumb broker*, como um cluster Kafka para centralização dos comandos vindos das bases de dados remotas de cada usuário, conforme proposto no diagrama de componentes na seção 4.3 deste documento. Outra melhoria também sugerida no diagrama de componentes deste documento é a utilização do docker para isolamento dos processos do sistema, facilitando a implantação e criação dos ambientes.

A segurança é tratada pelo CouchDB com as opções couch_peruser e require_valid_user que asseguram autenticação e autorização de leitura e escrita apenas na base de dados de propriedade do usuário.

A resiliência de rede e desempenho da aplicação cliente é assegurada pela abordagem *noBackend*, utilização da base dados local PouchDB e do padrão CQRS. Essa abordagem permite que todas as leituras sejam feitas dessa base local, garantido uma velocidade de busca muito alta nas tarefas principais do sistema (tela inicial – exibição/pesquisa de ofertas e tela de exibição/pesquisa de cupons).

A acessibilidade é tratada pelo Blazor (a implementação oficial da Microsoft para o WebAssembly, até o momento) com Bootstrap 4. Até este momento o Blazor está em release experimental e alguns de seus recursos podem não ter uma implementação tão flexível, porém para utilização nessa POC, seus recursos de renderização foram o suficiente para a abordagem SPA da aplicação cliente.

No geral a arquitetura apresenta mais pontos fortes do que limitações. Mostra-se ser uma arquitetura com possibilidade de crescimento do projeto e de fácil manutenção, possibilitando se integrar com outras aplicações.

7. Conclusão

Este trabalho apresentou um protótipo arquitetural de uma aplicação de dropshipping para uma aquisição de cupom promocional. Entende-se que os objetivos foram atingidos. Foram apresentadas algumas melhorias que não impactam a aceitação da proposta. Se houvesse mais tempo para o desenvolvimento elas seriam tratadas. Isso fica como sugestão para uma próxima versão.

REFERÊNCIAS

MICROSOFT AZURE. **Centro de Arquitetura do Azure**. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/> Acesso em: 08 de janeiro de 2019

APACHE COUCHDB. **Apache CouchDB 2.3 Documentation**. Disponível em: <https://docs.couchdb.org/en/stable/> Acesso em: 08 de janeiro de 2019

MICROSOFT AZURE. **Visão geral do Azure Service Fabric**. Disponível em: <https://docs.microsoft.com/pt-br/azure/service-fabric/service-fabric-overview> Acesso em: 08 de janeiro de 2019

OFFLINE CAMP. **Getting Started With Spiegel: Scalable Replication and Change Listening for CouchDB**. Disponível em: <https://medium.com/offline-camp/getting-started-with-spiegel-scalable-replication-and-change-listening-for-couchdb-8d9711ac29f8/> Acesso em: 08 de janeiro de 2019

BLAZOR. **Full-stack web development with C# and WebAssembly**. Disponível em: <https://blazor.net/> Acesso em: 08 de janeiro de 2019

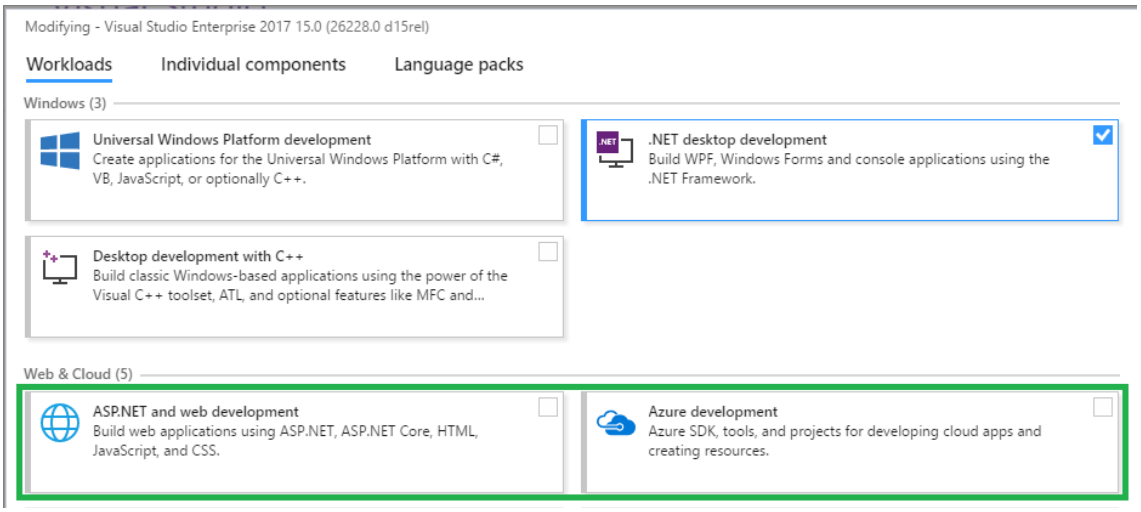
IRONPYTHON. **The Python programming language for the .NET Framework**. Disponível em: <http://ironpython.net/> Acesso em: 08 de janeiro de 2019

APÊNDICES

APÊNDICE A – Ambiente de Desenvolvimento

Os pré-requisitos da solução em ambientes de desenvolvimento são:

- Visual Studio 2017 (15.7 ou superior) Community com as workloads abaixo
  https://visualstudio.microsoft.com/pt-br/downloads/



- SDK do Microsoft Service Fabric
  https://docs.microsoft.com/pt-br/azure/service-fabric/service-fabric-get-started

- SDK do .NET Core 2.1
  https://www.microsoft.com/net/download/dotnet-core/2.1

- ASP.NET Core Blazor Languages Services
  https://marketplace.visualstudio.com/items?itemName=aspnet.blazor
- CouchDB
  https://dl.bintray.com/apache/couchdb/win/2.2.0/couchdb-2.2.0.msi

alterações:

[couchdb]
uuid = 4e5eee24d3d1dab4fb6d7d3ebeff8d19

[couch_peruser]
enable = true
delete_dbs = true

[chttpd]
require_valid_user = true

[httpd]
enable_cors = true

[couch_httpd_auth]
secret = 60ad9bb44eec09470e029e65850b1c39
require_valid_user = true
allow_persistent_cookies = true

[admins]
admin = -pbkdf2-38dd690c25bf24064ae626878c0508aeec7d5862,4747a24896b382f5c435c036bc8c68b4,10

[cors]
credentials = true
headers = accept, authorization, content-type, origin, referer
methods = GET, PUT, POST, HEAD, DELETE
origins = *

crie os databases:

`_users, _replicator and _global_changes`

user:

`user and pass podem ser vistos no arquivo:`
https://github.com/RenStu/Coupon/blob/master/Solution%20Itens/files.txt

- Node.js
https://nodejs.org/en/download/

execute:

```
npm install -g spiegel –unsafe
```

configurações:

`configurações da base spiegel podem ser vistos no arquivo:`
https://github.com/RenStu/Coupon/blob/master/Solution%20Itens/files.txt

```
- spiegel --type="install" --url="http://admin:passw0rd@localhost:5984"
- insert in spiegel database inside couchDB
        {
          "_id": "all_userdb_on_change",
          "type": "on_change",
          "db_name": "^userdb-.+",
          "if": {
                "type": "^command"
          },
          "url": "http://admin@localhost:8712/api/cmd",
          "params": {
                "Change": "$change",
                "DbName": "$db_name"
          },
          "method": "POST",
          "debounce": true
        }

- spiegel --type="update-listener" --url="http://admin:passw0rd@localhost:5984"
- spiegel --type="change-listener" --url="http://admin:passw0rd@localhost:5984" --
passwords-file="[FULL PATH]\change-listener-pass.json"
```

- Python 2.7
https://www.python.org/downloads/release/python-2715/

execute:

```
pip install beautifulsoup4
pip install request
```

APÊNDICE B – Código Fonte

O código fonte da prova de conceito que foi desenvolvido para atender os requisitos não funcionais dessa arquitetura de software.

URL do GitHub:
https://github.com/RenStu/Coupon/
URL da apresentação da POC no Youtube:
https://www.youtube.com/playlist?list=PLl9PrkTYv02j8p6P7C7VyNu_gDiqHHmK6