UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Características, padrões e soluções de desenvolvimento de aplicações com arquitetura de microserviços

Trabalho de Conclusão de Curso

João Paulo Feitosa Secundo

São Cristóvão – Sergipe

2022

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

João Paulo Feitosa Secundo

Características, padrões e soluções de desenvolvimento de aplicações com arquitetura de microserviços

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Rafael Oliveira Vasconcelos

São Cristóvão – Sergipe

2022

Lista de abreviaturas e siglas

API	Application Programing Interface - Interface para programação de aplicação
HTTP	HyperText Transfer Protocol - Protocolo de Transferência de HiperTexto
AMS	Arquitetura de MicroServiços

Sumário

1 Introdução

In recent years, the rise of the internet and the ubiquity of mobile computing have made it necessary for application developers to design their applications focusing on a lightweight, self-contained component.

Developers need to deploy applications quickly and make changes to the application without a complete redeployment. This has led to a new development paradigm called "microservices,"where an application is broken into a suite of small, independent units that perform their respective functions and communicate via APIs.

Although independent units, any number of these microservices may be pulled by the application to work together and achieve the desired results. (WHAT. . . ,)

1.1 Objetivos

1.1.1 Objetivo geral

Discutir, em alto nível, a arquitetura de microsserviços e suas características. Analisar os padrões, boas práticas, e soluções mais encontrados no desenvolvimento de aplicações que utilizam essa arquitetura. Modelar e implementar um exemplo de aplicação usando a arquitetura de microsserviços.

1.1.2 Objetivos específicos

- Caracterizar a arquitetura de microsserviços;

- Reunir padrões e práticas comuns na implementação de aplicações com arquitetura de microsserviços;

- Reunir soluções e ferramentas usadas na implementação de aplicações com arquitetura de microsserviços;

- Propor ideias e passos para como migrar do monolito para os microsserviços

Analisar a eficiência desses padrões e práticas.

Analisar a eficiência dessas soluções e ferramentas.

- Propor uma combinação desses padrões e dessas ferramentas para a construção de uma aplicação com arquitetura de microsserviços.

1.2 Metodologia

Para caracterizar a arquitetura de microsserviços, foram pesquisados as seguintes termos nas bases científicas ScienceDirect, SpringerLink e GoogleScholar:

- (microservices or microservice) and pattern

- (microservices or microservice) and provision

metodologia quanto aos objetivos, quanto a execução. passo a passo que vai seguir durante o trabalho.

explicar como analisar/testar essa eficiencia (objetivos especificos)

2

Fundamentação teórica

Introdução sobre as arquiteturas de monolito e de microsserviços. Trabalhos relacionados.

2.1 As aplicações monolíticas

Aplicações monolíticas são aplicações (não compostas?) que possuem as camadas de acesso aos dados, de regras de negócios, e de interface de usuário em um único programa em uma única plataforma. Os monolitos são autocontidos e totalmente independentes de outras aplicações. Eles são feitos não para uma tarefa em particular, mas sim para serem responsáveis por todo o processo para completar determinada função. Em outras palavras, as aplicações monolíticas têm problema de modularidade. Elas podem ser organizadas das mais variadas formas e fazer uso de padrões arquiteturais, mas são limitadas em muitos outros aspectos, citados na subseção 2.1.2.

2.1.1 Benefícios

O maior e melhor benefício da arquitetura monolítica é sua simplicidade. Uma aplicação simples é uma aplicação fácilmente entendida pelos seus desenvolvedores, o que melhora sua manutenibilidade. Para aplicações com um domínio simples, como um e-commerce de calçados por exemplo, optar por uma arquitetura complexa como a de microsserviços significaria adicionar uma enorme complexidade - provavelmente desnecessária - em seu desenvolvimento e infraestrutura.

Outra vantagem dos monolitos é sua facilidade de construção, tanto em relação a sua infraestrutura quanto ao seu desenvolvimento. Dentre todos os tipos de arquitetura, os monolitos têm o tipo de infraestrutura mais fácil de se construir, e além disso, nos monolitos geralmente não é necessário haver comunicação entre diferentes serviços ou máquinas, então os desenvolvedores não precisarão se preocupar com a complexidade que acompanha essa comunicação.

Até certo tamanho, são fáceis de manter porque são fáceis de serem entendidos. Porém,

depois de crescer excessivamente, um monolito pode se tornar um emaranhado complexo de funcionalidades que são difíceis de diferenciar, de separar, e de manter. E então começam a surgir as limitações deles...

2.1.2 Limitações

As limitações das aplicações monolíticas incluem crescimento, velocidade de desenvolvimento, confiabilidade, escalabilidade, manutenção, reutilização e flexibilidade.

Os problemas da solução monolítica incluem:

- A necessidade de compilar toda a aplicação, mesmo as partes em que não houve mudanças, a cada implantação.

- Falhas relativamente pequenas podem prejudicar toda a aplicação, mesmo as partes que não tiveram relação com a falha.

- As escolhas de tecnologias são mais limitadas. Um projeto tende a usar apenas 1 solução devido.

2.2 Os microsserviços

Aplicações com uma arquitetura de microsserviços são separadas em partes pequenas, chamadas de microsserviços, que são classificadas e se comunicam por meio de uma rede. Microsserviços oferecem capacidades de negócio ou de plataforma, tratando um aspecto em particular da aplicação. Eles se comunicam por meio de APIs bem definidas, contratos de dados, e configurações. O "micro"em microsserviços faz referência não ao tamanho do serviço, mas sim ao seu escopo de funcionalidade. Eles oferecem apenas uma determinada funcionalidade, tornando-se especialistas nela. Assim sendo, microsserviços não necessariamente devem ser pequenos em tamanho, mas fazem apenas uma tarefa e a fazem eficientemente.

Sendo especialistas em apenas uma tarefa, microsserviços têm características e comportamentos que os diferenciam de outras arquiteturas orientadas a serviços, os quais serão discutidos no Capítulo 3.

2.3 Trabalhos relacionados

"Microservices, IoT and Azure", por Bob Familiar - capítulo 2: "What is a microservice"

O capítulo 2 do livro de Bob Familiar descreve o que é um microsserviço, suas características e implicações, benefícios, e desafios.

"Microservices do one thing and they do it well". Como é explicado por Familiar (2015) , microsserviços representam business capabilities definidos usando o design orientado a domínio, são testados a cada passo do *pipeline* de *deploy*, e lançados por meio de automação, como serviços independentes, isolados, altamente escaláveis e resilientes em uma infraestrutura em núvem distribuída. Pertecem a um time único de desenvolvedores, que trata o desenvolvimento do microsserviço como um produto, entregando software de alta qualidade em um processo rápido e iterativo com envolvimento do cliente e satisfação como métrica de sucesso.

"A Systematic Mapping Study on Microservices Architecture in DevOps", por Waseem, M., Liang, P. e Shahin, M.

Esse trabalho tem o objetivo de sistematicamente identificar, analisar, e classificar a literatura sobre microsserviços em DevOps.

Inicialmente o leitor é contextualizado no mundo dos microsserviços e a cultura DevOps. Os autores usam a metodologia de pesquisa de um estudo de mapeamento sistemático da literatura publicada entre Janeiro de 2009 e Julho de 2018. Após selecionados 47 estudos, é feita a classificação deles de acordo com os critérios definidos pelos autores, e então é feita a discussão sobre os resultados obtidos - são expostos a quantidade de estudos sobre determinados tópicos em microsserviços, problemas e soluções, desafios, métodos de descrição, design patterns, benefícios, suporte a ferramentas, domínios, e implicações para pesquisadores e praticantes.

Os principais resultados são: (1) Três temas de pesquisa em AMS com DevOps são “desenvolvimento e operações de microsserviços em DevOps”, “abordagens e suporte a ferramentas para sistemas baseados em AMS em DevOps”, e “Experiência de migração de AMS em DevOps”. (2) São identificados 24 problemas e apontadas suas respectivas soluções com respeito a implementação de microsserviços com DevOps. (3) A AMS é descrita principalmente usando caixas e linhas. (4) A maioria das qualidades da AMS são afetadas positivamente quando aplicadas com DevOps. (5) 50 ferramentas que suportam a construção de sistemas baseados em AMS são apontados. (6) A combinação da AMS e DevOps tem sido aplicada em uma ampla variedade de domínios de aplicações.

(Comparar cada trabalho com o meu trabalho. Coisas que eles não abordam e que eu abordo)

3
Características

3.1 Propriedades

3.1.1 Autonomia e Isolamento

Autonomia e isolamento significa que microsserviços são unidades auto-contidas de funcionalidade com dependências de outros serviços fracamente acopladas e são projetados, desenvolvidos, testados e lançados independentemente. (FAMILIAR, 2015). O termo autônomo pode ser definido como - existe ou é capaz de existir independetemente das outras partes. O termo isolado, como - separado das outras partes.

3.1.2 Elasticidade, resiliência, e responsividade

Microsserviços são reusados entre muitas soluções diferentes e portanto devem ser escaláveis de acordo com o uso. Devem ser resilientes, isso é, ser tolerantes a falhas e ter um tempo de recuperação razoável quando algo der errado. Além disso, devem ser responsivos, tendo um desempenho razoável de acordo com o uso. (FAMILIAR, 2015) O termo elástico pode ser definido como - capaz de retornar ao tamanho/formato original depois de ser esticado, comprimido ou expandido. O termo resiliente, como - resistente às mudanças negativas. O termo responsivo, como - Rápido em responder e reagir.

3.1.3 Orientação-a-mensagens e programabilidade

Microsserviços dependem de APIs e contratos de dados para definir como interagir com o serviço. A API define um conjunto de endpoints acessíveis por rede, e o contrato de dados define a estrutura da mensagem que é enviada ou retornada. (FAMILIAR, 2015). O termo orientado-a-mensagens pode ser definido como - Software que conecta sistemas separados em

uma rede, carregando e distribuindo mensagens entre eles. O termo programável, como - Obedece a um plano de tarefas que são executadas para alcançar um objetivo específico.

3.1.4 Configurabilidade

Microsserviços devem provêr mais do que apenas uma API e um contrato de dados. Para que seja reusável e para que possa resolver as necessidades do sistema que o use, cada microsserviço tem níveis diferentes de configuração, e esta configuração pode ser feita de diferentes formas. (FAMILIAR, 2015). O termo configurável pode ser definido como - Projetado ou adaptado para formar uma configuração ou para algum propósito.

3.1.5 Automação

O ciclo de vida de um microsserviço deve ser totalmente automatizado, desde o design até a implantação. O termo automatizado pode ser definido como - Funcionar sem precisar ser controlado diretamente.

3.2 Vantagens

3.2.1 Evolução

Quanto maior e mais antigo o software, mais difícil é de dar manutenção, e monolitos envelhecem com maior velocidade do que microsserviços. Entretanto, é possível migrar de um sistema monolítico para a arquitetura de microsserviços aos poucos, um serviço por vez, identificando capacidades de negócio, implementando-as como um microsserviço, e integrando com uso de padrões de baixo acoplamento. Ao longo do tempo, mais e mais funcionalidades podem ser separadas e implementadas como microsserviço, até que o núcleo da aplicação monolítica se transforme em apenas um outro serviço, ou um microsserviço. (FAMILIAR, 2015)

3.2.2 Possibilidade de uso de diferentes ferramentas

Cada microsserviço disponibiliza suas funcionalidades por meio de APIs e contratos de dados em uma rede. A comunicação independe da arquitetura que o microsserviço faz uso, então cada microsserviço pode escolher seu sistema operacional, linguagem e banco de dados. Isso é especialmente valioso para times com dificuldade de comunicação, pois cada time precisa apenas de conhecimento da arquitetura do microsserviço em que trabalha. (FAMILIAR, 2015)

3.2.3 Alta velocidade

Com um time responsável por cuidar do ciclo de desenvolvimento e sua automação, a velocidade com que microsserviços podem ser desenvolvidos é muito maior do que fazer o

equivalente para uma solução monolítica. (FAMILIAR, 2015)

3.2.4 Reusável e combinável

Microsserviços são reusáveis por natureza. Eles são entidades independentes que provêm funcionalidades em um determinado escopo por meio de padrões de internet aberta. Para criar soluções para o usuário final, multiplos microsserviços podem ser combinados. (FAMILIAR, 2015)

3.2.5 Flexível

O *deploy* de microsserviços é definido por sua automação. Essa automação pode incluir configuração de cenários diferentes de uso, não apenas para produção, mas também para desenvolvimento e testagem, possibilitando que o microsserviço tenha o melhor desempenho em diversos cenários. Para tanto é necessário uso de ferramentas que configurem essa flexibilidade. Tais ferramentas serão melhor discutidas no Capítulo 5 (FAMILIAR, 2015)

3.2.6 Versionável e Substituível

Com o controle completo dos cenários de implantação, é possível manter versões diferentes de um mesmo serviço rodando ao mesmo tempo, proporcionando retrocompatibilidade e fácil migração. Além disso, serviços podem ser atualizados ou mesmo substituidos sem ocasionar indisponibilidade do serviço. (FAMILIAR, 2015)

3.3 Desafios

3.3.1 [re]Organizaçao

Organizar o sistema e o time para sustentar uma arquitetura de microsserviços é um grande desafio. Como explica Familiar (2015):

> If you are part of a command-and-control organization using a waterfall software project management approach, you will struggle because you are not oriented to high-velocity product development. If you lack a DevOps culture and there is no collaboration between development and operations to automate the deployment pipeline, you will struggle. (FAMILIAR, 2015)

3.3.2 Plataforma

Criar o ambiente de execução para microsserviços requer um grande investimento em infraestrutura dinâmica em *data centers* dispersos para garantir maior disponibilidade. Se sua atual plataforma *on-premises* não suporta automação, infraestrutura dinâmica, escalamento

elástico e alta disponibilidade, deve-se considerar uma plataforma na núvem. Mais sobre soluções na núvem será discutido no Capítulo 5.

3.4 Identificação

Domain-driven design (design orientado a domínio) é uma tecnica bem consolidada e muito usada. Mas para aplica-la em microsserviços, é preciso analisar onde cada peça deve ficar. Em vez de projetar os modelos e os (contextos limitados) separando-os em camadas, pode-se juntar os contextos com seus respectivos modelos, e procurar por possíveis pontos de (separação) da aplicação - um lugar onde a linguagem muda, por exemplo. Isso resultaria em um ponto de partida para separar as partes de uma arquitetura de microsserviços.

3.5 Testes

Assim como em qualquer aplicação, o teste é uma parte crucial do seu desenvolvimento. Escrever e testar código não muda muito entre as arquiteturas monolítica e de microsserviços, contudo, nos microserviços existem mais testes a serem executados. Não deve-se testar o microsserviço apenas antes de seu lançamento, mas sim em cada passo do *pipeline* de implantação, sempre automatizando o máximo de etapas possível, para assim garantir uma entrega rápida de software de qualidade.

3.6 Descoberta

Encontrar microsserviços em um ambiente distribuido pode ser feito de algumas maneiras diferentes. A informação pode ser armazenada diretamente no código, pode ser guardada e acessada em um arquivo, ou pode ser construido um microsserviço para encontrar outros microsserviços e disponibilizar suas localizações. Contudo, para prover detectabilidade como um serviço será necessário adquirir um produto de terceiros, integrar um projeto aberto, ou desenvolver sua própria solução.

4

Padrões

4.1 Antes de tudo, o monólito

> But as with any architectural decision there are trade-offs. In particular with microservices there are serious consequences for operations, who now have to handle an ecosystem of small services rather than a single, well-defined monolith. Consequently if you don't have certain baseline competencies, you shouldn't consider using the microservice style. (FOWLER, 2014)

Fowler (2014) afirma que existem 3 pré-requisitos para se adotar uma arquitetura de microserviços, e que na grande maioria dos casos, deve-se começar pela arquitetura monolítica até que o sistema já esteja bem definido. Os pré-requisitos são - provisionamento rápido, monitoramento básico e deploy rápido de aplicação.

4.1.1 Provisionamento rápido

No contexto da computação, provisionamento significa disponibilizar um recurso, como uma máquina virtual por exemplo. Para produzir software, é necessário provisionar muitos recursos, tanto para os desenvolvedores quanto para o cliente. Naturalmente, o provisionamento é mais fácil na núvem. Na AWS por exemplo, para conseguir uma nova máquina, basta lançar uma nova instância e acessá-la - um processo muito rápido quando comparado ao *on-premises*, onde precisaria-se comprar uma nova máquina, esperar chegar, configurá-la, e só então ela estará pronta. Para alcançar um provisionamento rápido, será necessário bastante automação.

4.1.2 Monitoramento básico

Muitas coisas podem dar errado em qualquer tipo de arquitetura, mas em especial nos microserviços pois cada serviço é fracamente acoplado, estando sujeitos não só a falhas no código, mas também na comunicação, na conexão, ou até falhas físicas. Portanto o monitoramento é

crucial nesse tipo de arquitetura para que problemas, especialmente os mais graves possam ser detectados no menor tempo possível. Além disso, o monitoramento também pode ser usado para detectar problemas de negócio, como uma redução nos pedidos por exemplo.

4.1.3 Deploy rápido de aplicação

Na arquitetura de microserviços o deploy geralmente é feito separadamente para cada microserviço. Com muitos serviços para gerenciar, o deploy pode se tornar uma tarefa árdua, portanto será novamente necessário uma automação dessa etapa, que geralmente envolve um pipeline de deploy, que deve ser automatizado o máximo possível.

4.2 Configuração

4.3 Deploy

4.4 Comunicação entre microserviços

4.5 Testes

Mas além dos métodos mais conhecidos de testes, como test-driven development, teste de unidade e teste funcional, é necessário testar os microsserviços conforme passam pelo *pipeline* de *deploy*.

- Testes internos: Testar as funções internas do serviço, inclusive uso de acesso de dados, e caching.

- Teste de serviço: Testar a a implementação de serviço da API. Essa é uma implementação privada da API e seus modelos associados.

- Teste de protocolo: Testar o serviço no nível de protocolo, chamando a API sobre o determinado protocolo (geralmente HTTP).

- Composition Testing: Test the service in collaboration with other services within the context of a solution.

- Scalability/Throughput Testing: Test the scalability and elasticity of the deployed microservice.

- Failover/Fault Tolerance Testing: Test the ability of the microservice to recover after a failure.

- PEN Testing: Work with a third-party software security firm to perform penetration testing. NOTE: This will requires cooperation with Microsoft if you are pen testing microservices deployed to Azure.

4.6 Como migrar do monolito para os microserviços

4.6.1 Identificação

If you are currently working with a complex layered architecture and have a reasonable domain model defined, the domain model will provide a roadmap to an evolutionary approach to migrating to a microservice architecture. If a domain model does not exist, you can apply domain-driven design in reverse to identify the bounded contexts, the capabilities within the system. (FAMILIAR, 2015)

4.6.2 Organização

Em uma mudança do monolito para microsserviços, é recomendado que não sejam feitas mudanças grandes e abruptas na sua organização. Em vez disso, deve-se procurar uma oportunidade com uma iniciativa de negócio para testar a fórmula proposta por Familiar (2015) :

- Formar um pequeno time inderdisciplinar (cross-functional?).

- Oferecer treinamento e orientação na adoção de práticas ágeis, como o scrum.

- Oferecer uma localização física separada para esse time trabalhar a fim de não afetá-lo negativamente por politicas internas ou hábitos antigos.

- Adotar uma abordagem de minimo produto viável para entregar pequenos mas incrementais *releases* de software, usando essa abordagem durante todo o ciclo de vida.

- Integrar esse serviço com sistemas existentes, usando um acoplamento solto.

- Percorrer esse ciclo de vida do microsserviço diversas vezes, fazendo as adaptações necessárias até chegar a equipe ficar confortável com o processo.

- Colocar o time principal em posições de liderança enquanto são formados novos times interdisciplinares para disseminar o conhecimento e a prática.

5
Soluções

5.1 Flexibilidade

(tais como as ferramentas de Auto Scaling da AWS)

5.2 Plataforma

Microsoft Azure is a microservice platform, and it provides a fully automated dynamic infrastructure, SDKs, and runtime containers along with a large portfolio of existing microservices that you can leverage, such as DocumentDb, Redis In-Memory Cache, and Service Bus, to build your own microservices catalog. (FAMILIAR, 2015)

6
Conclusão

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Plano de Continuidade

Analisar a eficiência desses padrões e práticas.

Analisar a eficiência dessas soluções e ferramentas.

Propor uma combinação desses padrões e dessas ferramentas para a construção de uma aplicação com arquitetura de microsserviços.

Referências

FAMILIAR, B. What is a microservice? In: ____. *Microservices, IoT, and Azure: Leveraging DevOps and Microservice Architecture to Deliver SaaS Solutions*. Berkeley, CA: Apress, 2015. p. 9–19. ISBN 978-1-4842-1275-2. Disponível em: <https://doi.org/10.1007/978-1-4842-1275-2_2>. Citado 6 vezes nas páginas 9, 10, 11, 12, 16 e 17.

FOWLER, M. *bliki: MicroservicePrerequisites*. 2014. Blog do Martin Fowler. Disponível em: <https://martinfowler.com/bliki/MicroservicePrerequisites.html>. Acesso em: 06 Oct 2022. Citado na página 14.

WHAT are Microservices? How Microservices architecture works. Disponível em: <https://middleware.io/blog/microservices-architecture/>. Citado na página 5.

Apêndices

APÊNDICE A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APÊNDICE B – Nullam elementum urna vel imperdiet sodales elit ipsum pharetra ligula ac pretium ante justo a nulla curabitur tristique arcu eu metus

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Anexos

ANEXO A – Morbi ultrices rutrum lorem.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Cras non urna sed feugiat cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANEXO C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.