



UNIPÊ
Centro Universitário
de João Pessoa

Especialização em
Desenvolvimento Web Full Stack

FUNDAMENTOS E ARQUITETURA DE APLICAÇÕES WEB

PROGRAMAÇÃO DE HOJE



INTRODUÇÃO

Conceitos de Web
e Arquitetura
Web



CONCEITOS

DESENVOLVIMENTO
WEB e Seus Ambientes



PRÁTICA

Desenvolvimento
Web e
Versionamento de
Código

VAMOS NOS CONHECER MELHOR?



Prof. Daniel Brandão

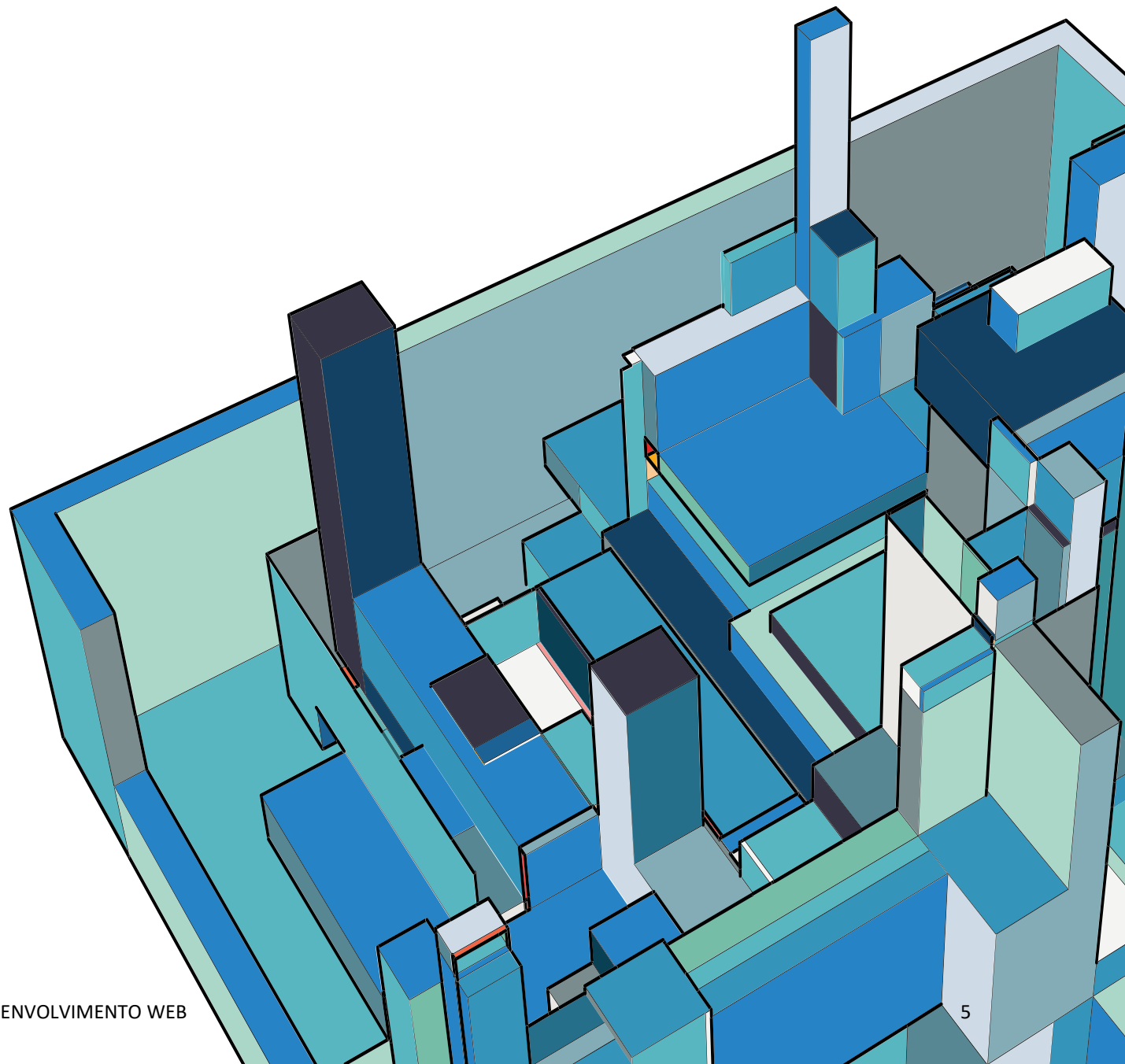


Mestrando em TI. Especialista em Tecnologias Web. Graduado em Sistemas para Internet.

- Professor em Graduação e Pós na área de Programação, Banco de dados, Análise e Visualização de Dados.
- Gerente de Projetos de TI na **SES-PB**.
- Instrutor e Consultor em Dados pela **FABWORK**.
- Pesquisador e desenvolvedor na **FORD**.



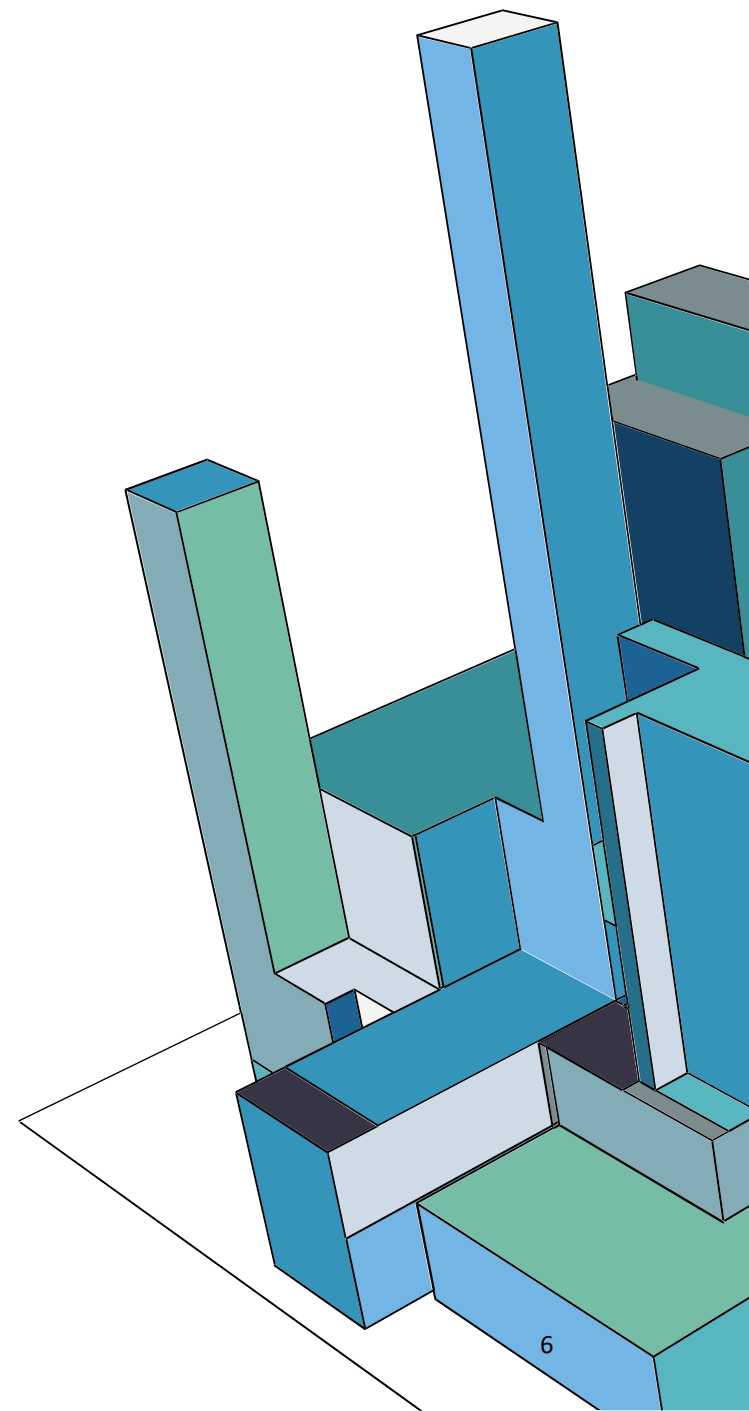
WWW



WWW

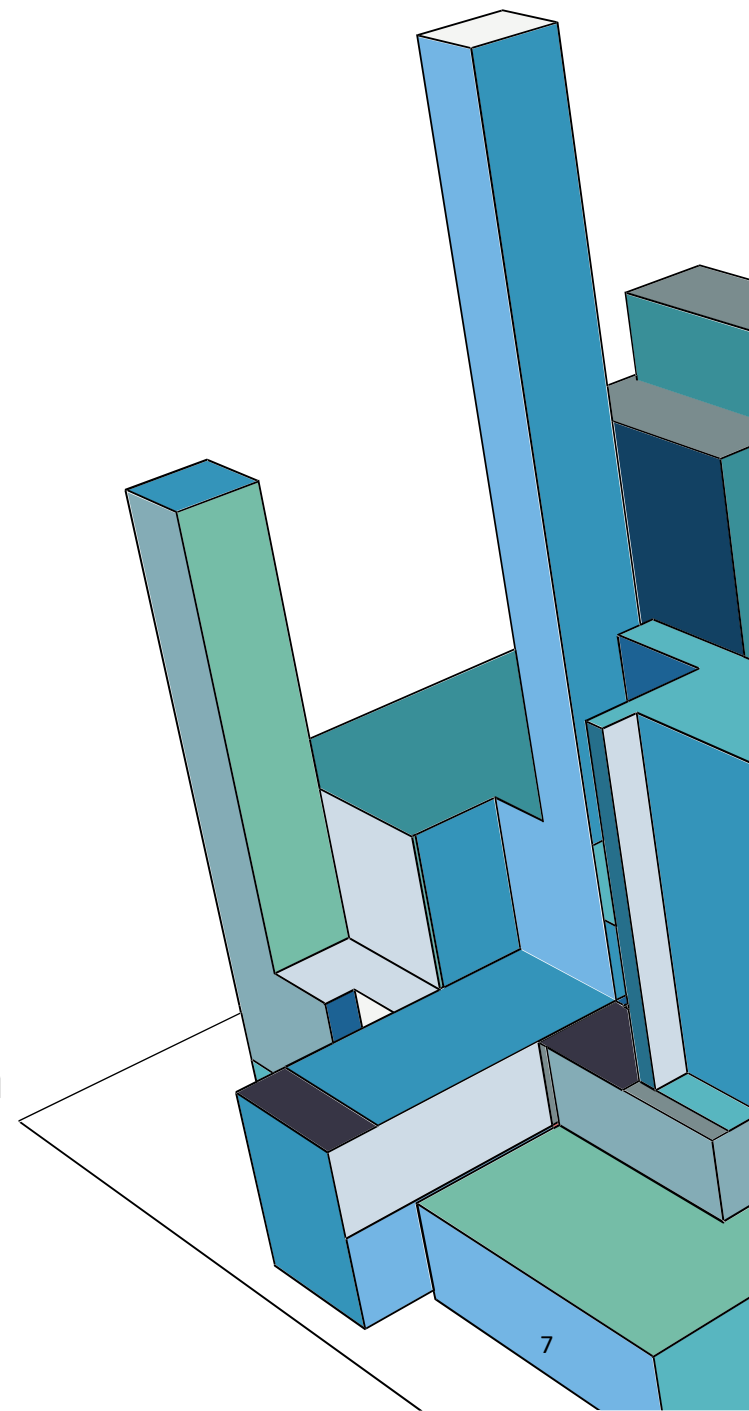
“Durante muito tempo, a definição mais aceita de **Web** é de algo que tem uma **URL**. Essa é a visão do ‘sistema de informação da web’. Vem sendo percebido que os três pilares da Web são: identificadores, formatos e protocolos. As URLs são as mais universais e estáveis.”

Mark Nottingham - presidente do Grupo de Trabalho do **HTTP** na **IETF (Internet Engineering Task Force)**



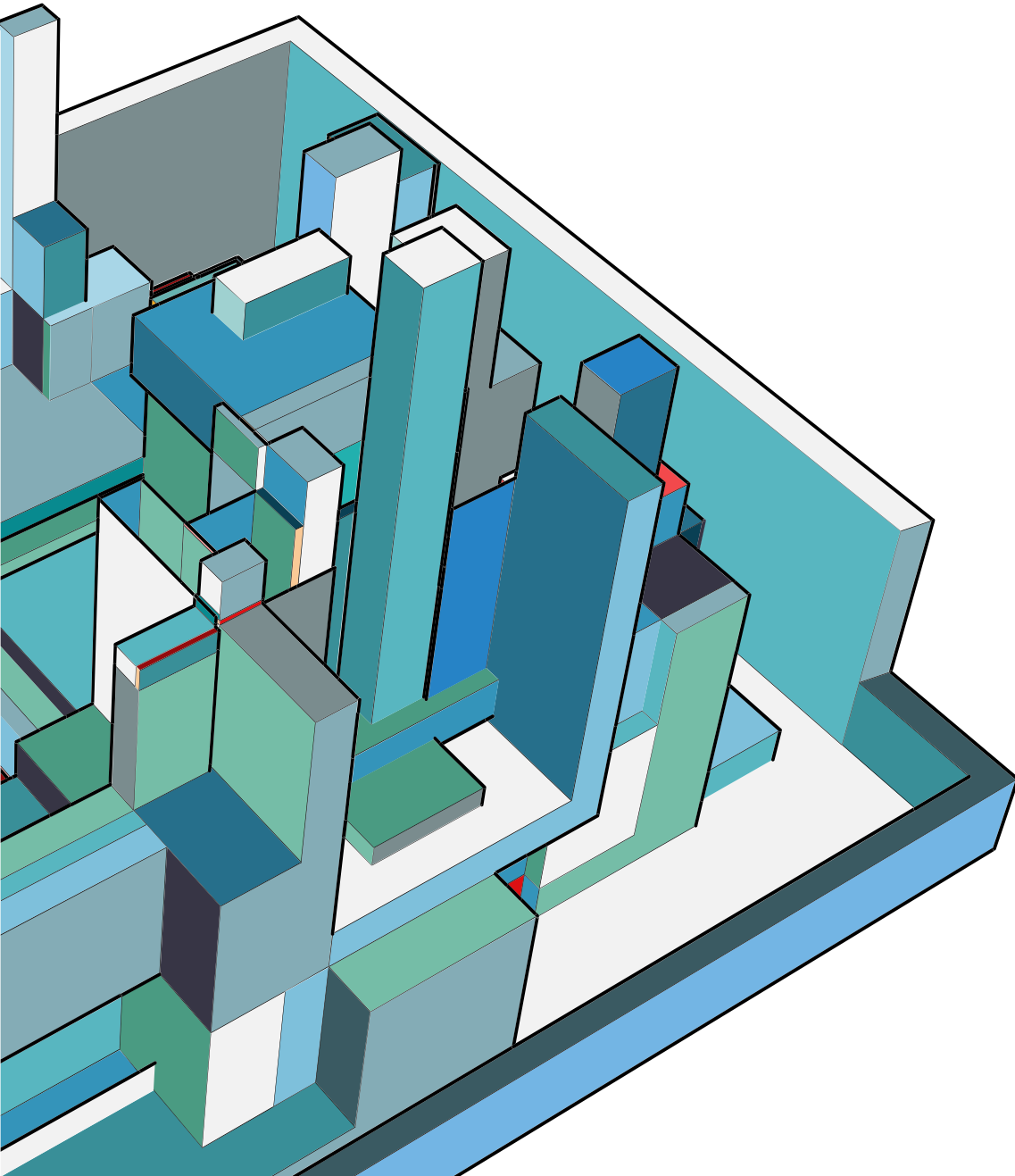
WWW

- Em 1992, o cientista **Tim Berners-Lee** criou a World Wide Web;
- A rede nasceu na Organização Europeia para a Investigação Nuclear (CERN em inglês);
- Propôs a criação dos **hipertextos** para permitir que várias pessoas trabalhassem juntas acessando os mesmos documentos. Esta foi a gênese do processo de conexão à Internet atual.



World Wide Web

Surgiu, então, no começo dos anos 80 o primeiro BOOM da web. Empresas como **GOOGLE** e **NETSCAPE** foram as pioneiras. No Brasil tivemos **BOL** e **UOL**



The classic Google logo from the late 1990s, featuring the word "Google!" in its signature multi-colored font (blue, red, yellow, blue, green, red) with a slight 3D effect and a shadow.

Search the web using Google!

10 results

Google Search

I'm feeling lucky

Index contains ~25 million pages (soon to be much bigger)

[About Google!](#)

[Stanford Search](#) [Linux Search](#)

Get Google! updates monthly!

your e-mail

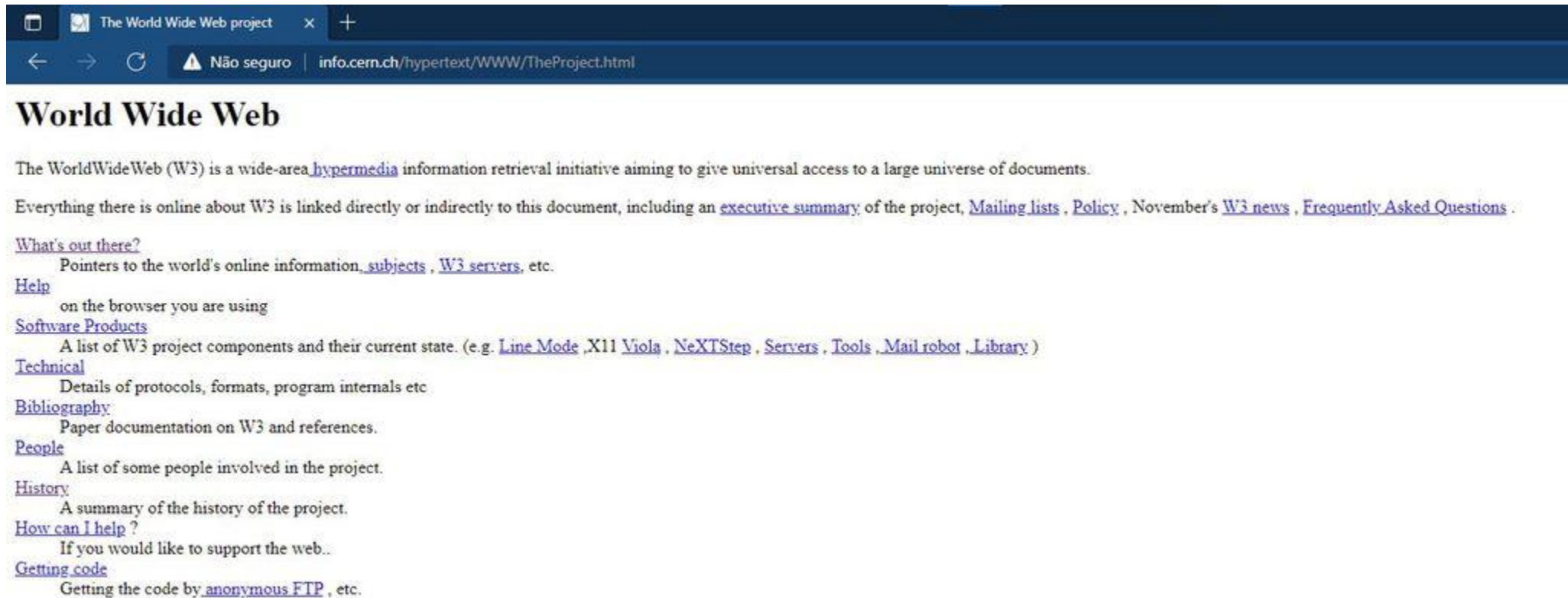
Subscribe

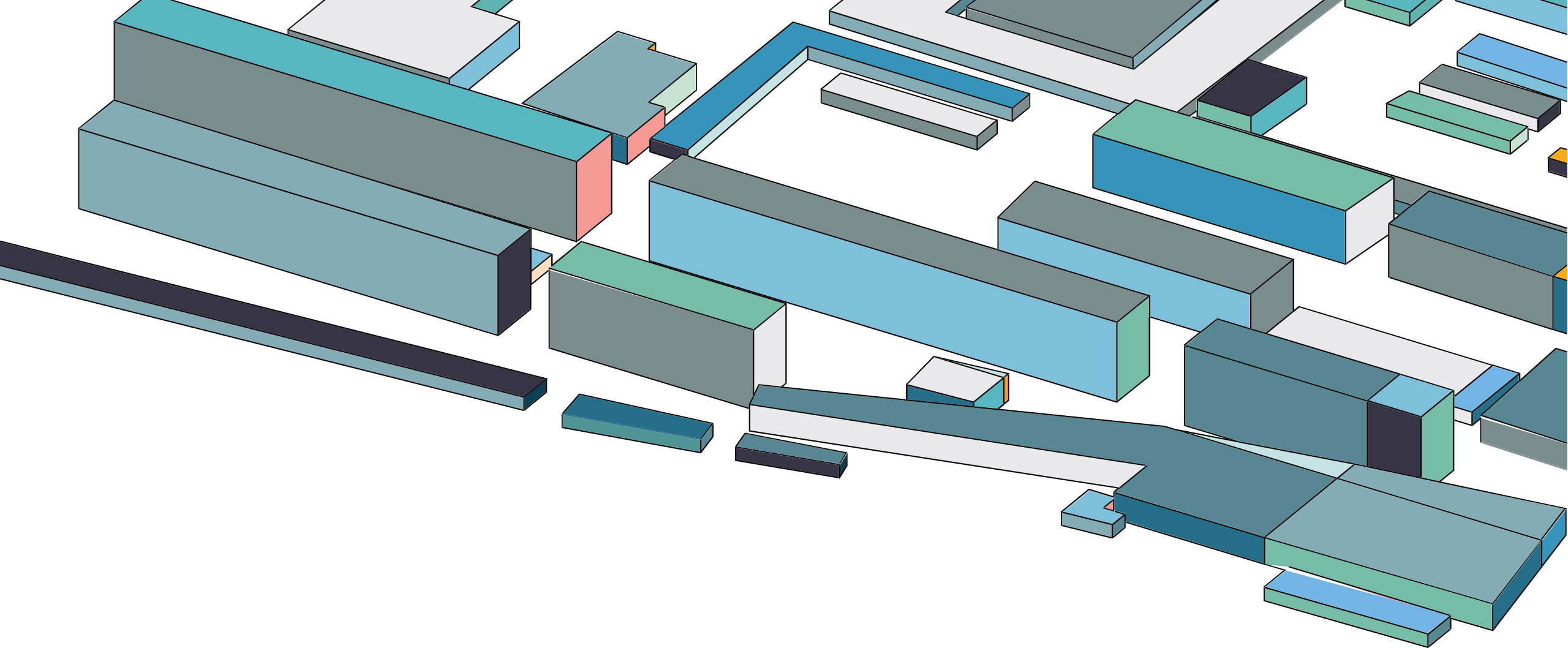
[Archive](#)

Copyright ©1997-8 Stanford University

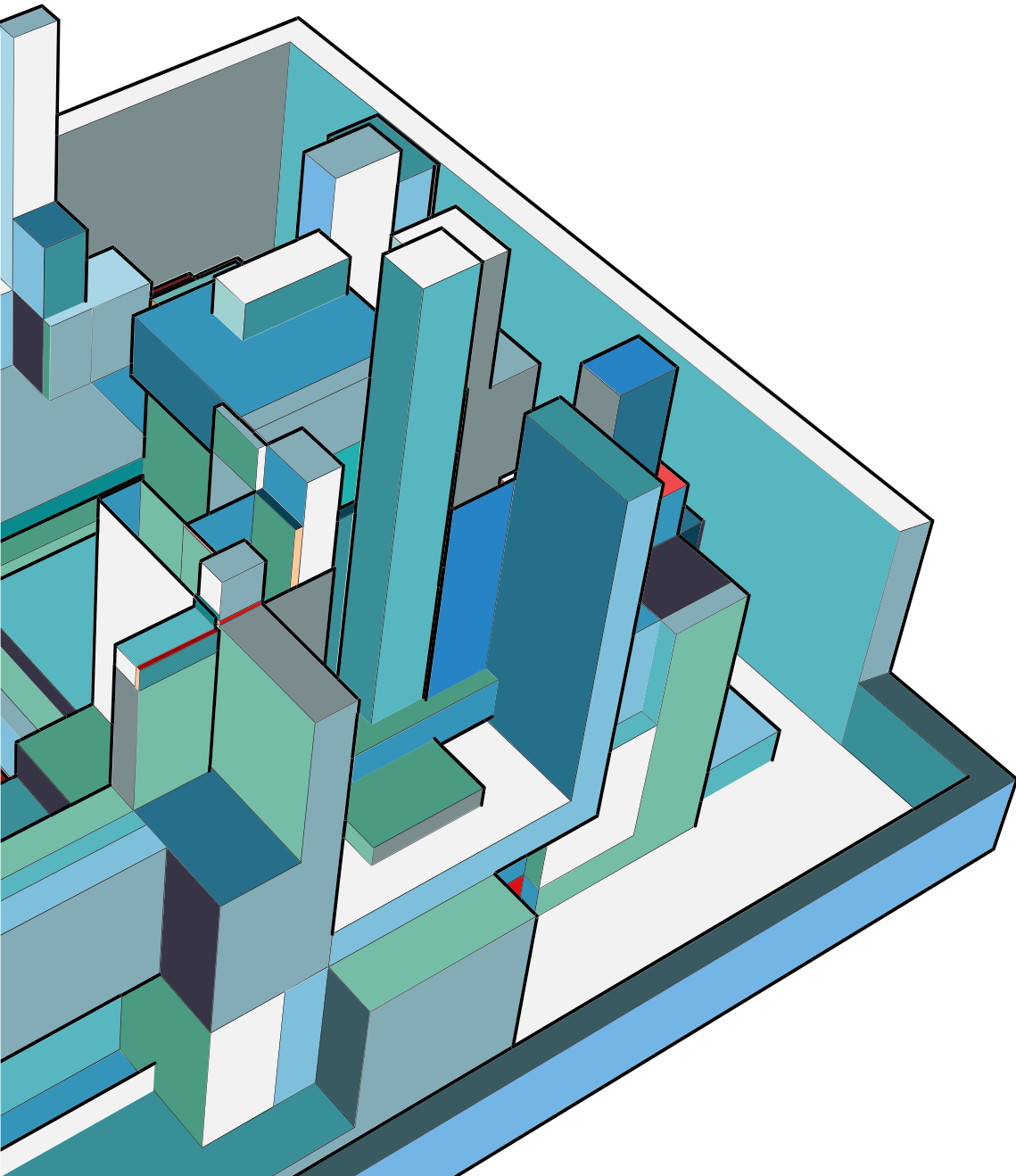
DE

THE PROJECT – 1º Site da História





ARQUITETURA WEB



ARQUITETURA WEB

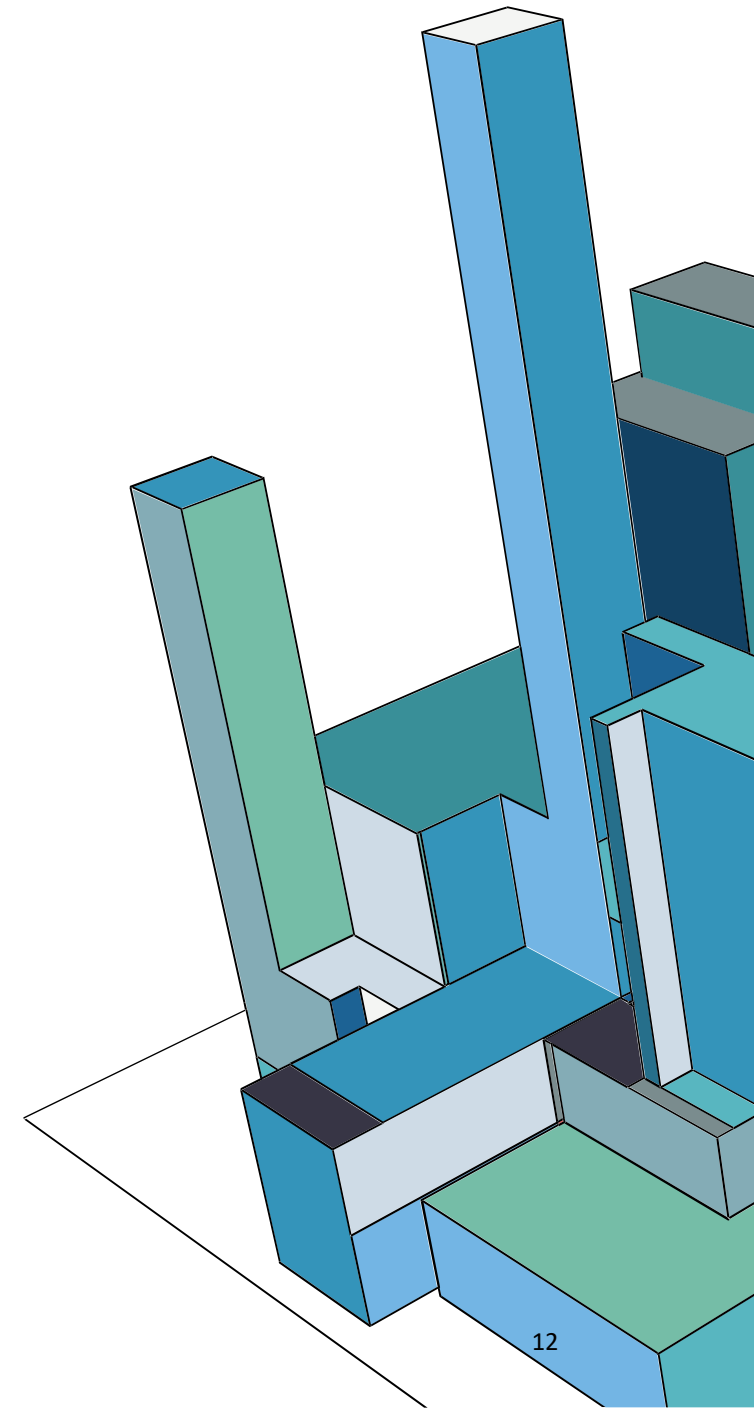
PADRÕES DE ARQUITETURA WEB visam orientar os desenvolvedores quanto a adoção de boas práticas no processo de desenvolvimento de suas aplicações, expressando a forma de organizar a estrutura fundamental de uma aplicação.

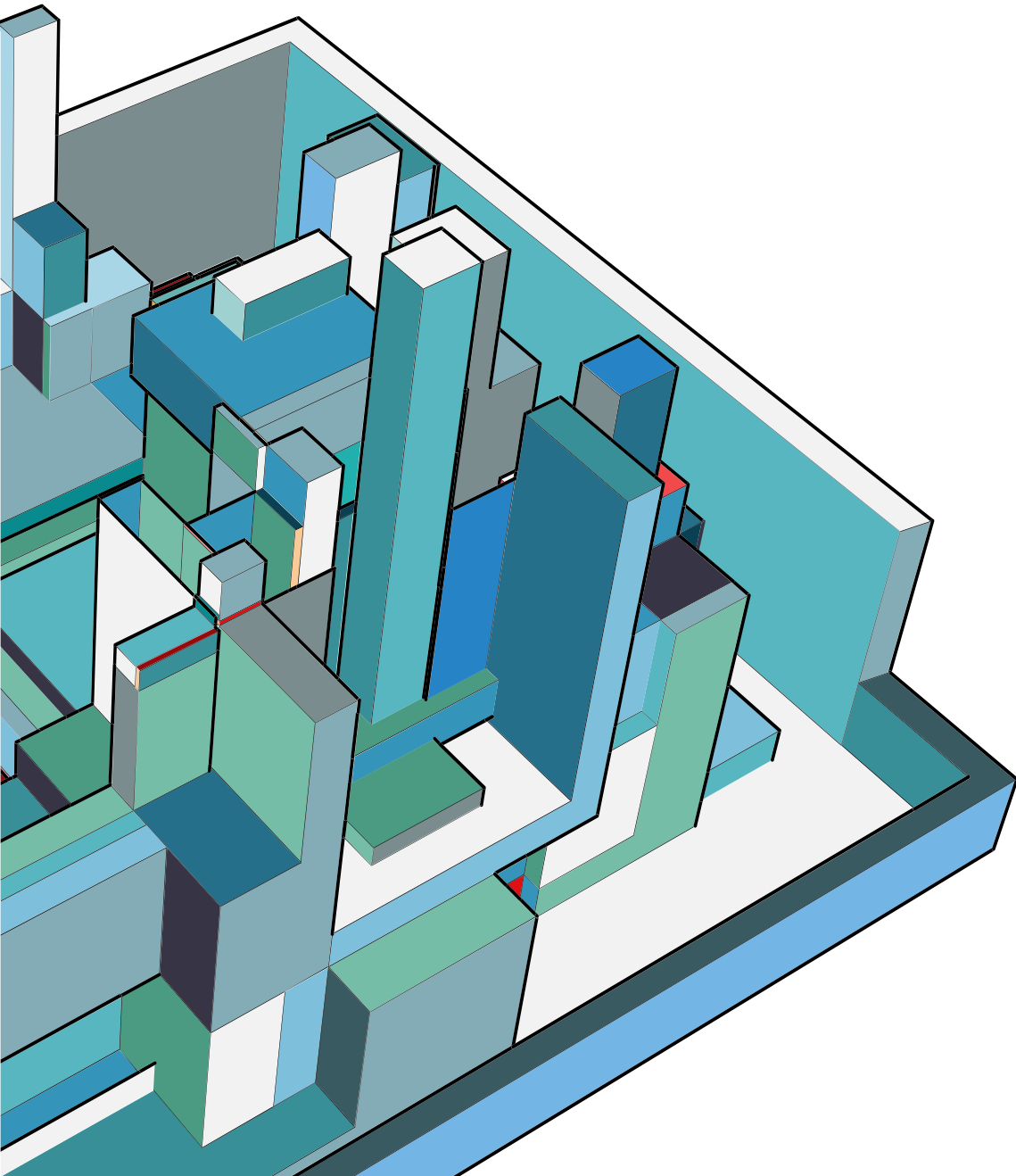
Arquiteturas de Sistemas Web

A arquitetura de software cliente-servidor não é mais a única suficiente. Desse modo, este modelo precisou unir forças com outros modelos formando arquiteturas cada vez mais heterogêneas para atender uma demanda crescente e exigente.

Padrões de arquitetura

- **Monolíticas:** componentes agrupados em um único sistema.
- **Microserviços:** aplicação dividida em vários serviços independentes.

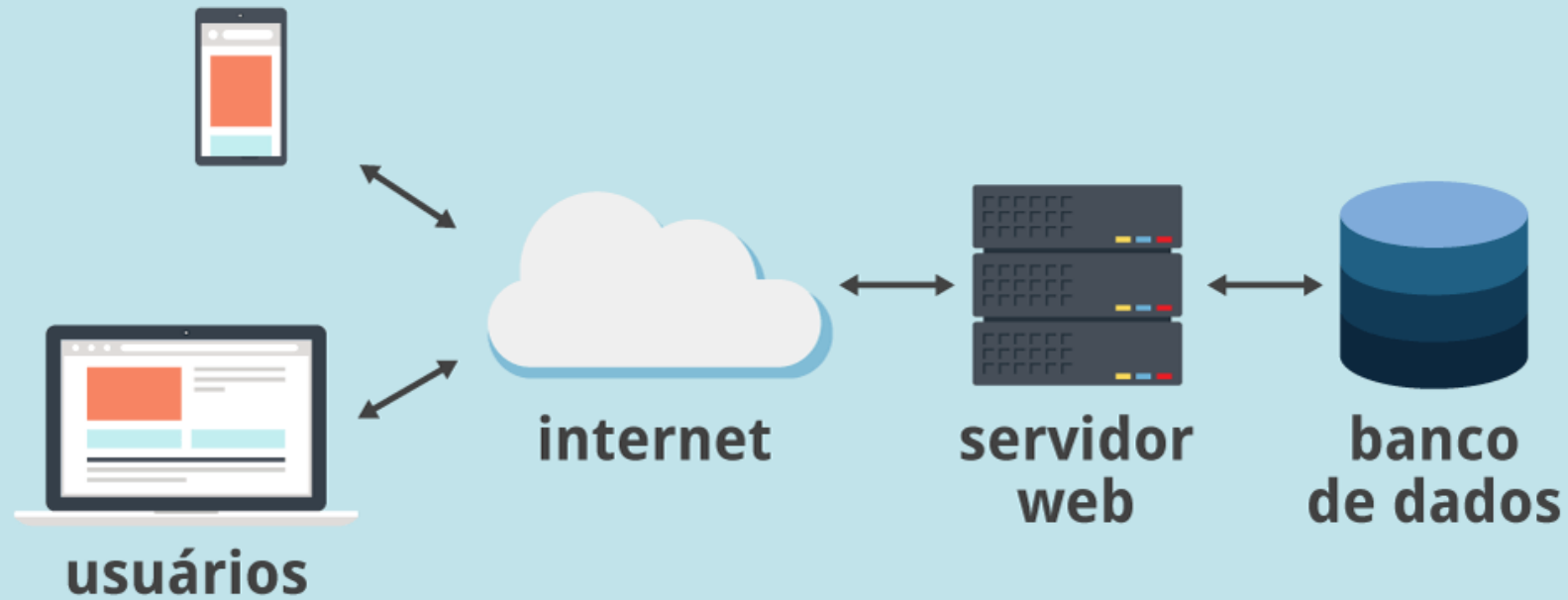




ARQUITETURA WEB

Com o surgimento de novas tecnologias, novas linguagens de programação, frameworks, protocolos, regras de negócio cada vez mais complexas, os padrões de arquitetura também precisaram se reinventar.

Arquiteturas de Sistemas em Camadas



Arquiteturas de Sistemas em Camadas

Camada	Responsabilidades
Apresentação	Fornecimento de serviços; Exibição de informações; Tratamento de solicitações do usuário;
Lógica	O real propósito do sistema; Ponte entre as camadas de apresentação e dados; A camada “inteligente” do sistema;
Camada de Dados	Comunicação com bancos de dados; Armazenamento de dados persistentes;



Arquiteturas de Sistemas Web

As três principais camadas

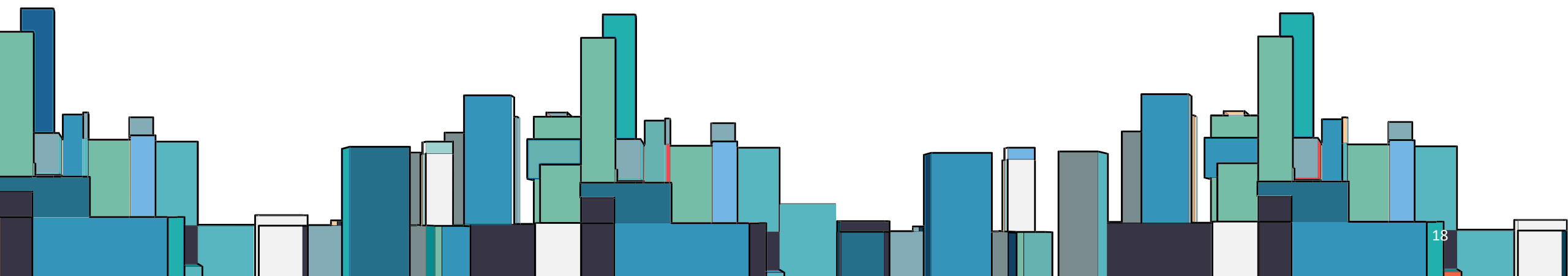
- Cada camada resolve e cuida de problemas específicos;
- Cada camada envolve um componente;
- Cada componente é facilmente identificável;
- As camadas podem ser implementadas em um ou mais computadores/servidores;
- A separação e o uso de camadas irá depender da complexidade da aplicação em questão;

Arquiteturas de Sistemas Web

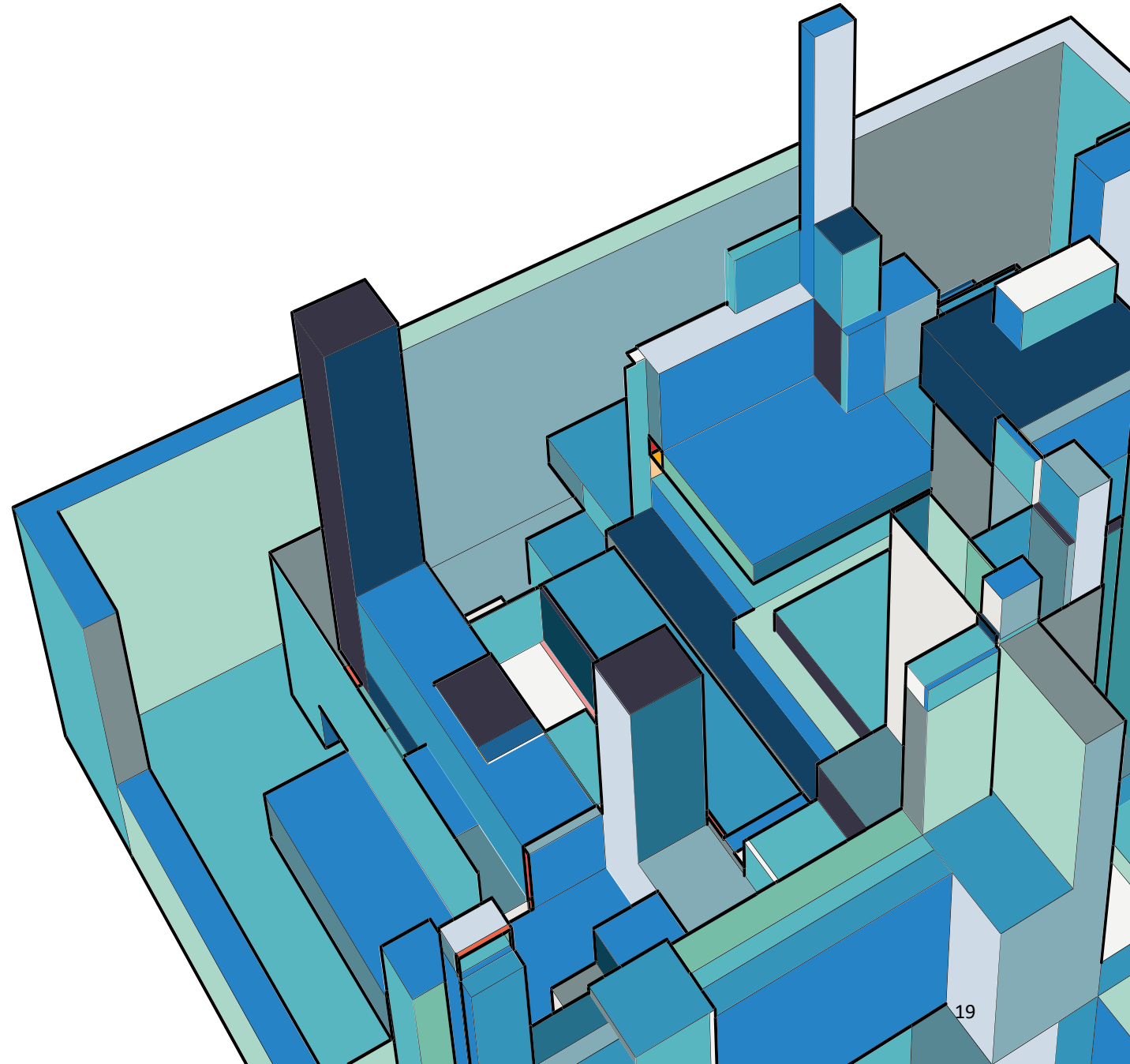
Camada	Tecnologias
Apresentação	HTML CSS Javascript
Lógica	PHP, Java, Python etc.
Gerenciamento de Dados	MySQL PostgreSQL...

Características dos padrões de arquitetura web

- **Flexibilidade:** os padrões de arquitetura web devem ser flexíveis o suficiente para atender às diferentes necessidades das aplicações web.
- **Escalabilidade:** os padrões de arquitetura web devem ser escaláveis para suportar o crescimento do tráfego e da demanda.
- **Manutenção:** os padrões de arquitetura web devem ser fáceis de manter e evoluir.
- **Segurança:** os padrões de arquitetura web devem proteger os dados da aplicação de acessos não autorizados.

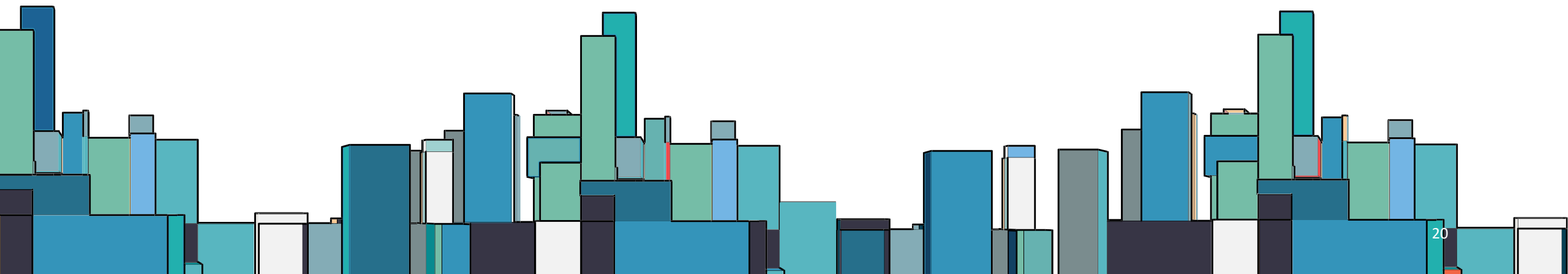


Arquitetura Orientada a Microserviços



Arquitetura de Microserviços

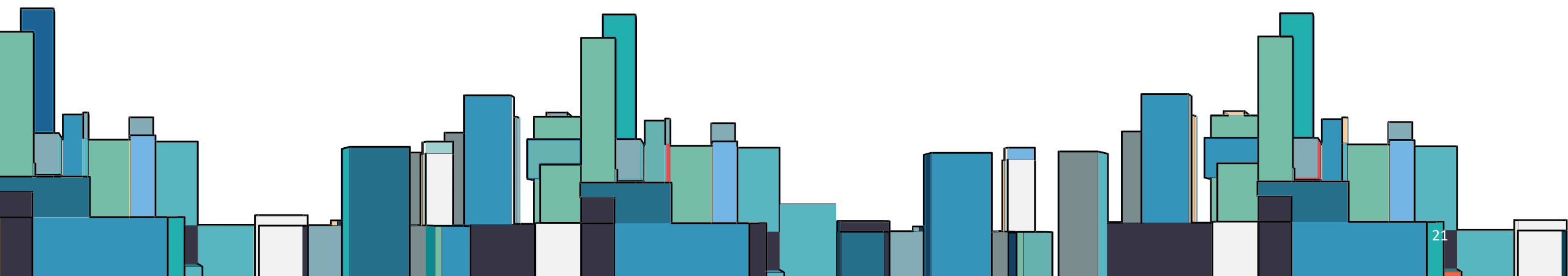
Microserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs bem definidas. Esses serviços pertencem a pequenas equipes autossuficientes.



Arquitetura de Microserviços

Diferenças entre as arquiteturas monolítica e de microserviços

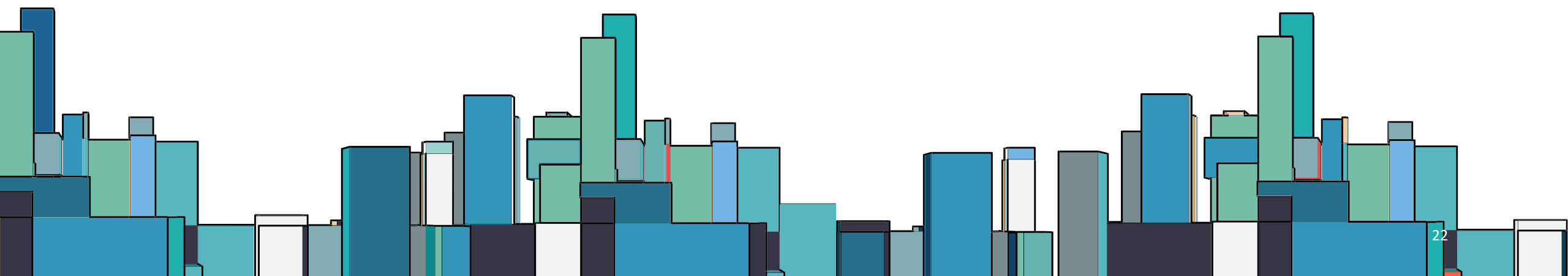
Com as arquiteturas monolíticas, todos os processos são altamente **acoplados** e executam como um único serviço. Isso significa que se um processo do aplicativo apresentar um pico de demanda, toda a arquitetura deverá ser escalada. A complexidade da adição ou do aprimoramento de recursos de aplicativos monolíticos aumenta com o crescimento da base de código.



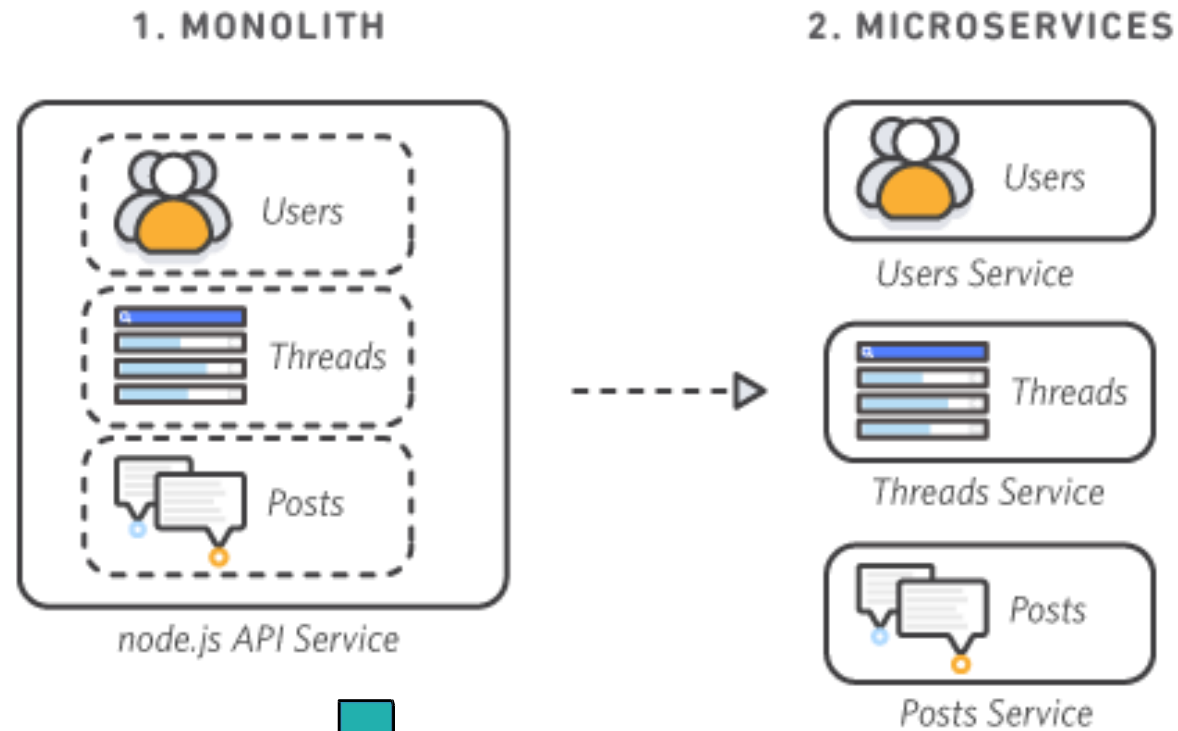
Arquitetura de Microserviços

Diferenças entre as arquiteturas monolítica e de microserviços

Essa complexidade limita a experimentação e dificulta a implementação de novas ideias. As arquiteturas **monolíticas** aumentam o risco de disponibilidade de aplicativos, pois muitos processos dependentes e altamente acoplados aumentam o impacto da falha de um único processo.



Arquitetura de Microserviços



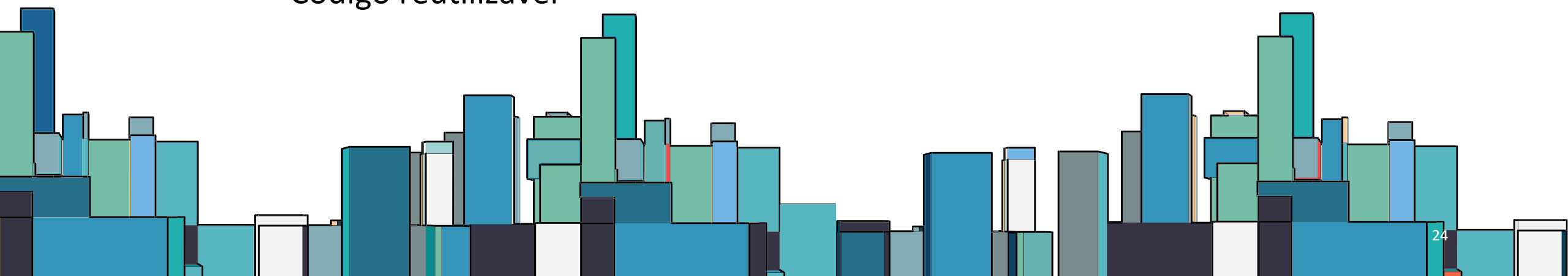
Arquitetura de Microserviços

Principais Vantagens

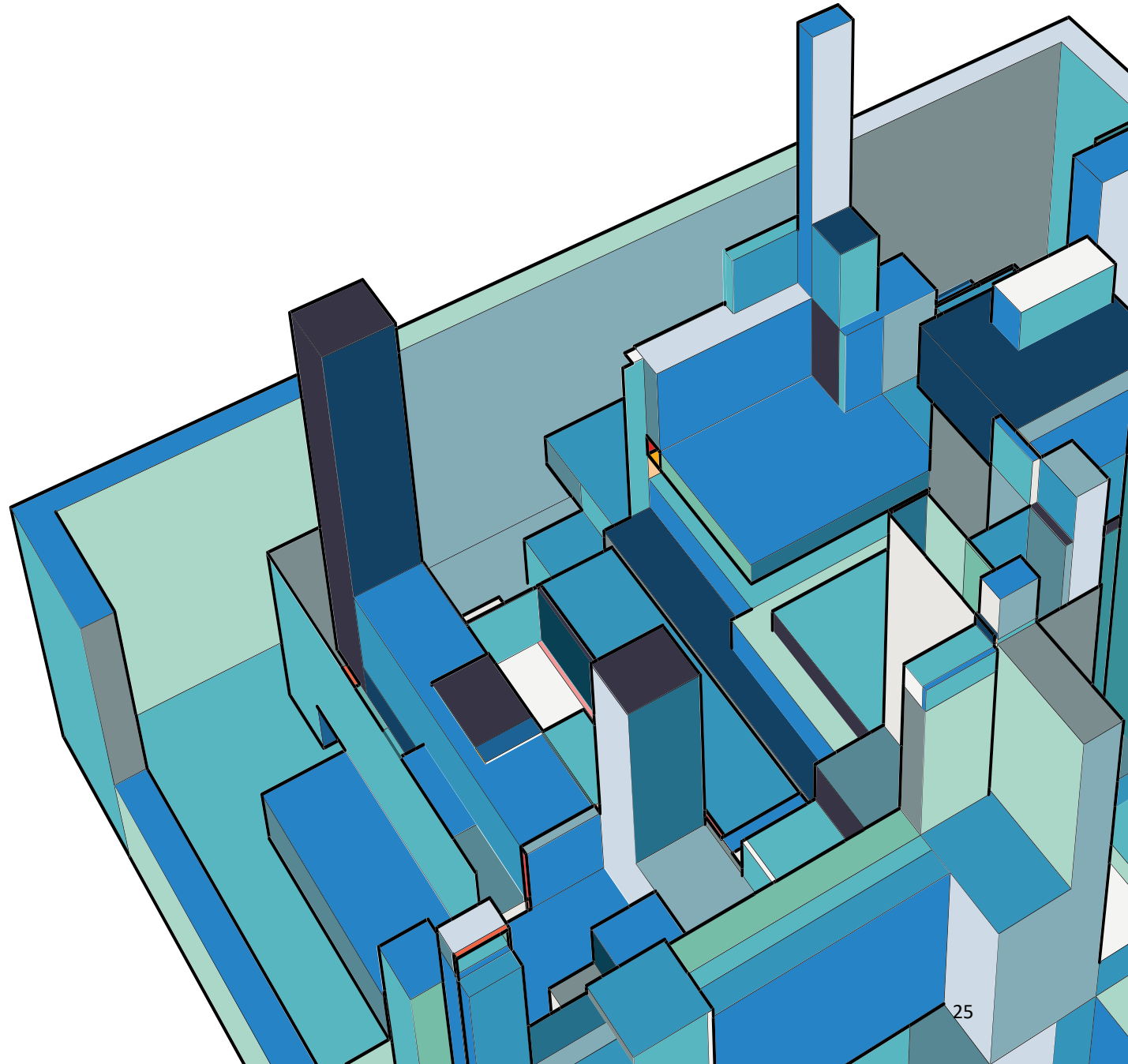
- Agilidade
- Escalabilidade flexível
- Fácil implantação
- Liberdade tecnológica
- Código reutilizável

Principais Desvantagens

- Complexidade de infraestrutura e recursos
- Comunicação entre microserviços
- Segurança
- Testes
- Gerenciamento complexo

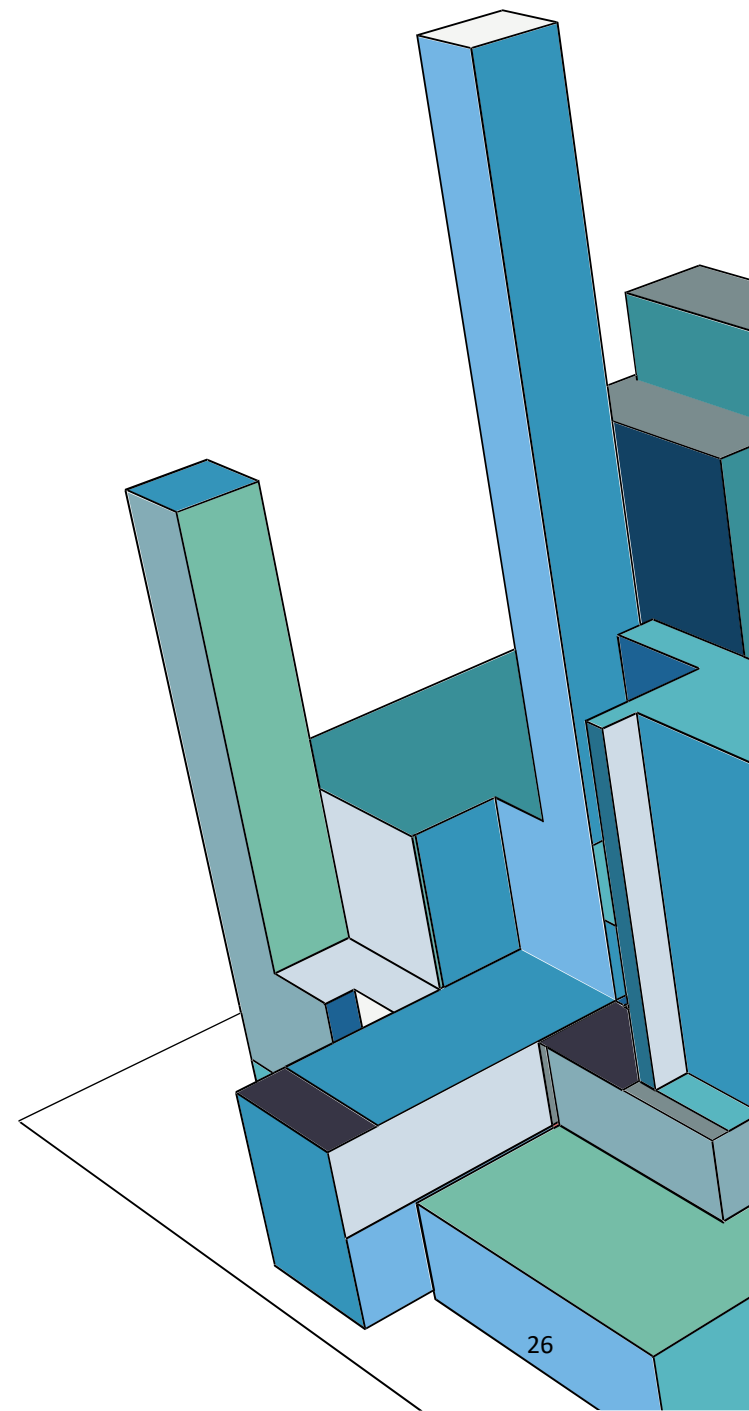


PROTOS



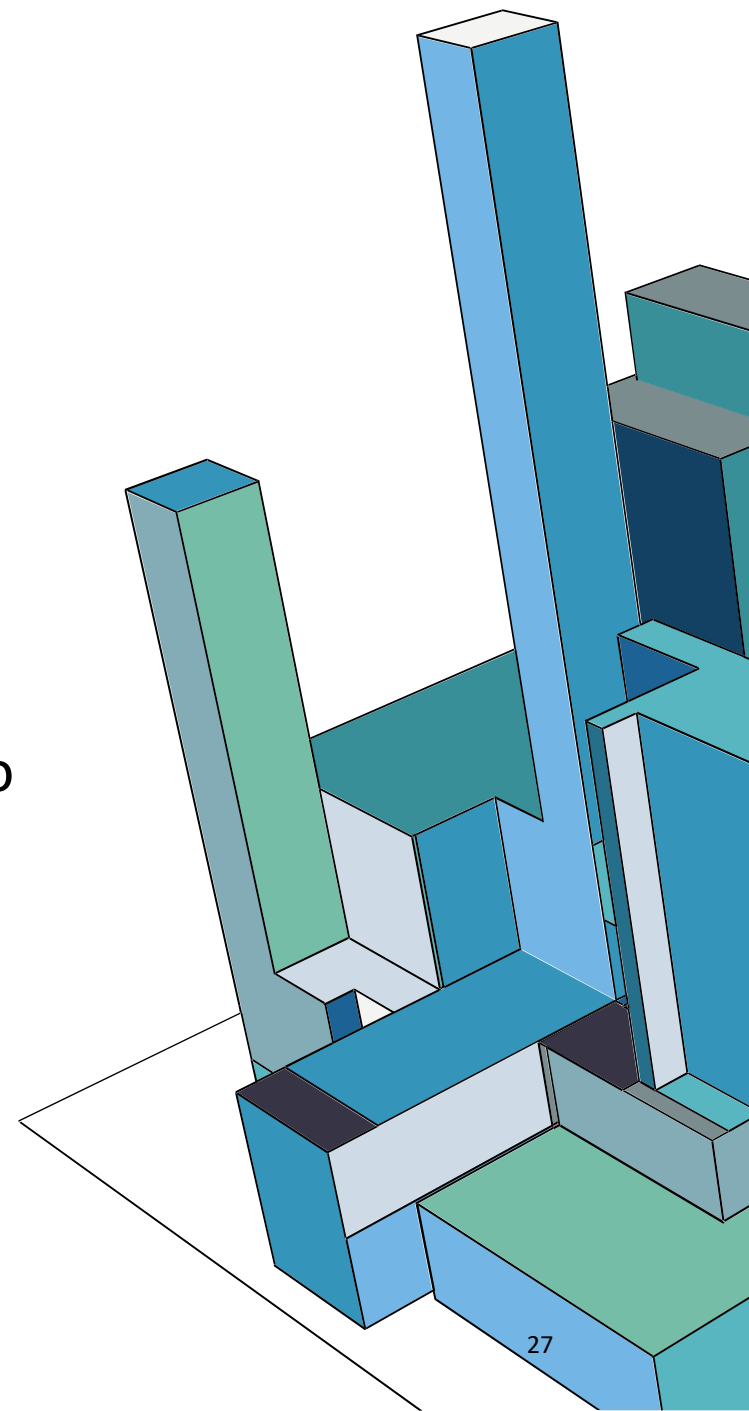
PROTOCOLOS

- Convenção ou padrão;
- Controla e possibilita uma conexão, comunicação ou transferência de dados entre dois sistemas operacionais;
- Na Web, a forma como podemos trocar mensagens;

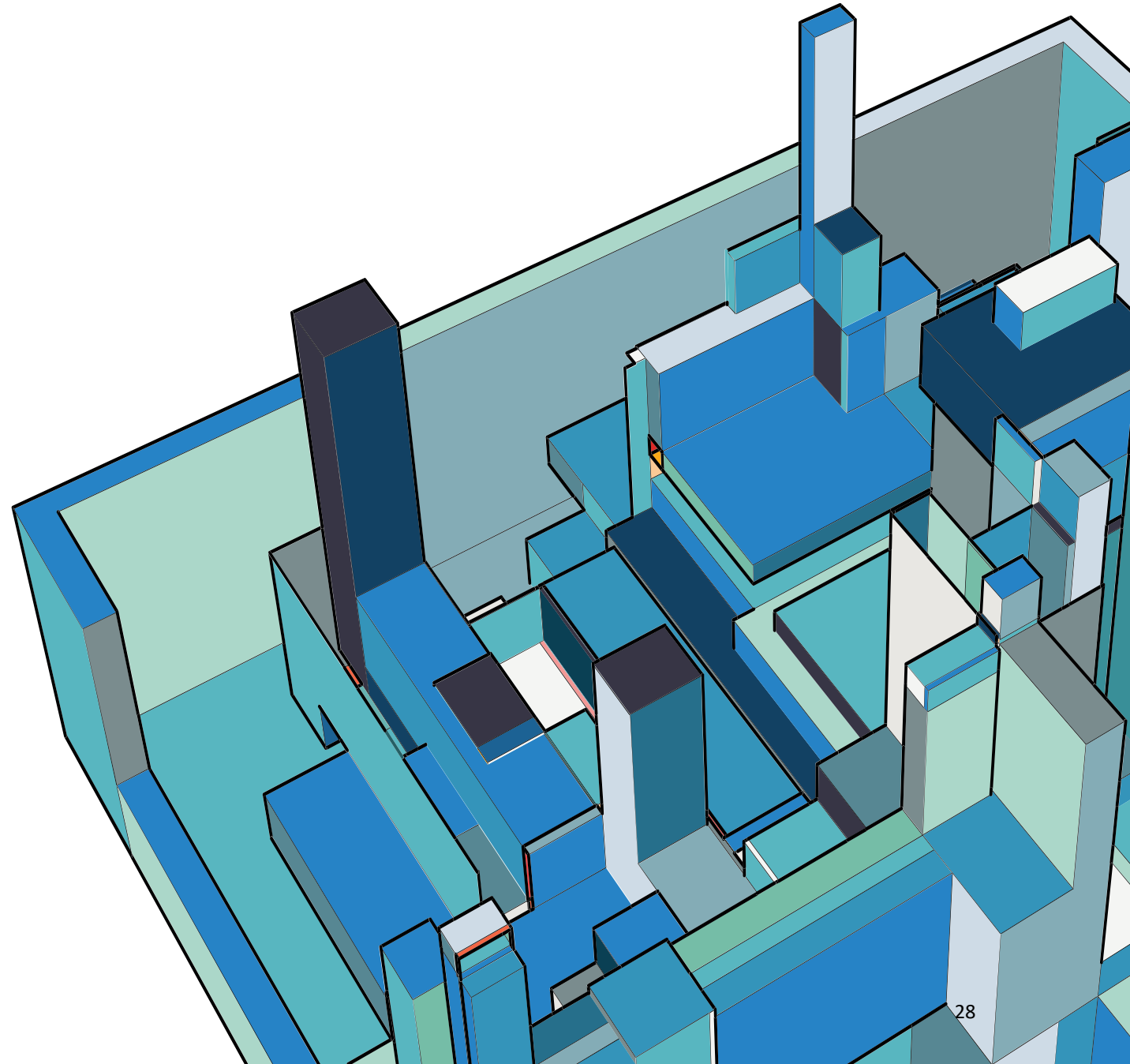


PROTOCOLOS

- Exemplos de protocolo:
 - **HTTP** (Hypertext Transfer Protocol) – Protocolo de transferência de hipertexto;
 - **FTP** (File Transfer Protocol) – Protocolo de transferência de arquivos;
 - **POP** (Post Office Protocol) – Protocolo de acesso remoto a uma caixa de e-mail;
 - **SMTP** (Simple Mail Transfer Protocol) – Protocolo de envio de email simples

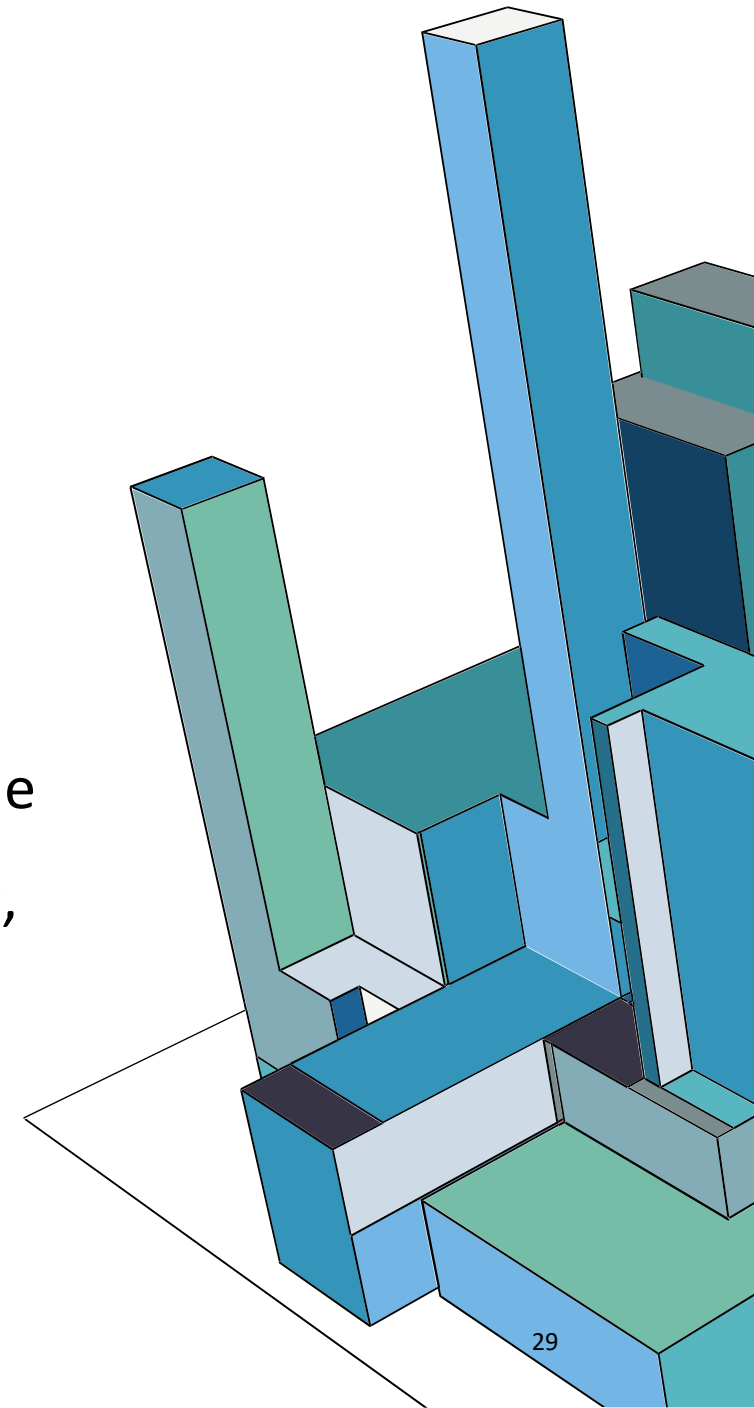


CONTROLE DE VERSÃO



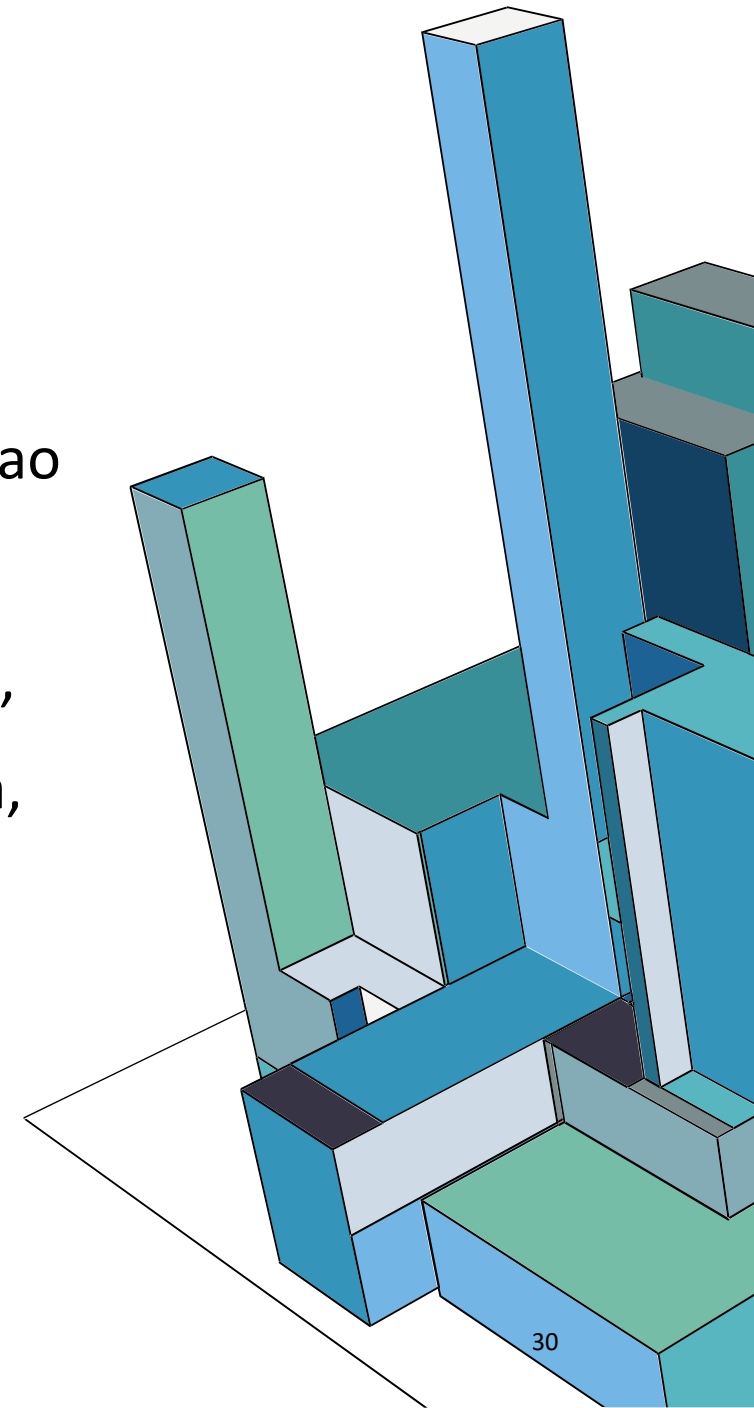
O que é controle de versão?

- O **controle de versão(CV)** é um sistema usado para ter controle sobre todas (isso se for usada corretamente) as mudanças feitas em um determinado arquivo.
- Como exemplo, CV permite você reverter sua aplicação que se encontra em um estado que está apresentando um bug, para um estado anterior onde o bug não havia se manifestado.



O que é controle de versão?

- CV permite você descobrir quem introduziu um problema ao repositório, quando e onde foi introduzido.
- Além disso, quando estiver usando um repositório remoto, não correrá o risco de perder seu arquivos e, melhor ainda, você também não perderá o controle sobre as mudanças feitas localmente.





GIT

- O **Git** é a ferramenta que utilizamos para fazer todo o controle de versão.
- Surgiu quando Linus Torvalds, o criador do Linux, começou a enfrentar problemas com as ferramentas de versionamento da época, quando desenvolvia o kernel do Linux (projeto open-source que ele trabalhava com apoio de uma comunidade) sentindo a necessidade da criação de uma nova ferramenta.



GIT

O Git é, portanto, um projeto de código aberto maduro e com manutenção ativa desenvolvido em 2005, cuja proposta era apresentar algumas features que sistemas antigos não ofereciam, como:

1. Velocidade;
2. Simplicidade;
3. Forte suporte para desenvolvimento não-linear (milhares de ramos paralelos);
4. Completamente distribuído;
5. Capaz de lidar com grandes projetos com velocidade e grande tamanho dos dados.

CLIENTES PARA WINDOWS



PARA LINUX | PARA Mac OS

- Git Force
- Qgit
- GitG

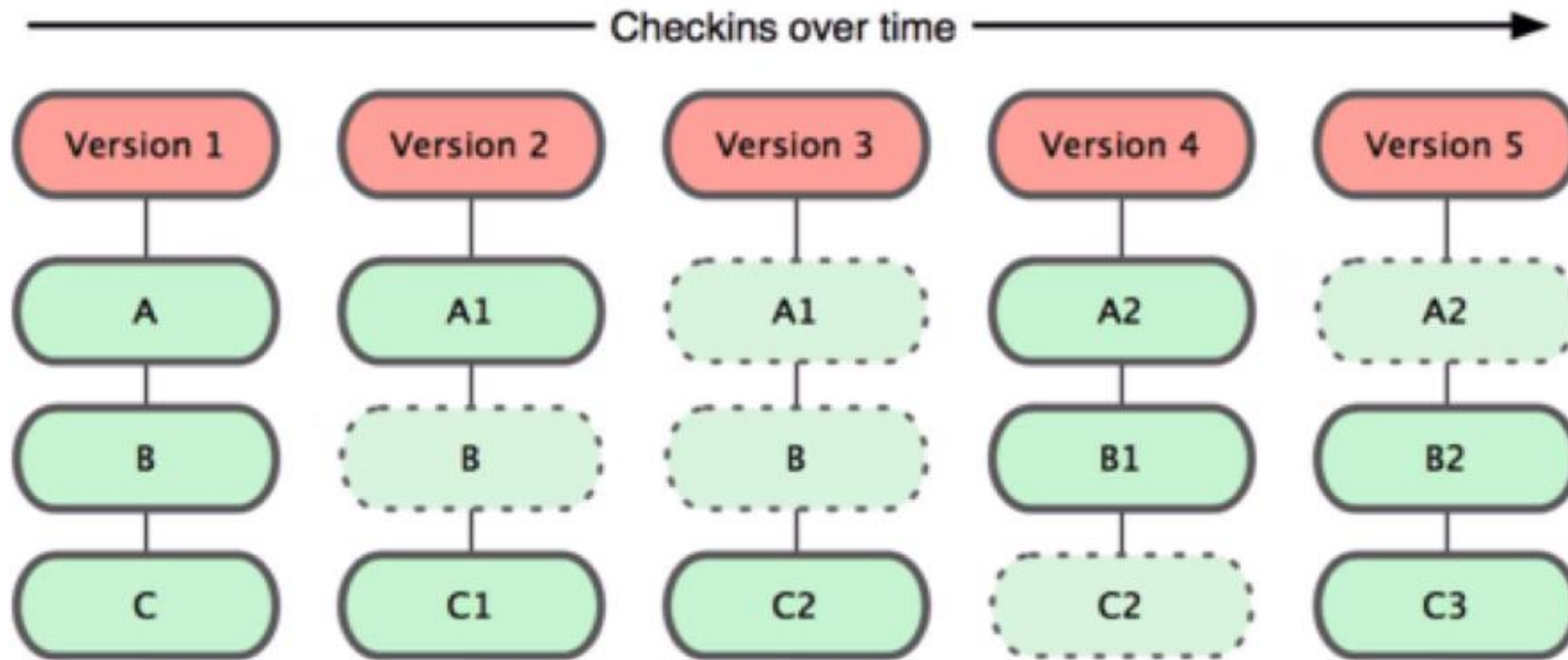
- GitUp
- GitBox
- GitX-Dev



GITHUB

- Mais famosos repositório online, o GitHub é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git.
- Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo.

VERSIONAMENTO NO SISTEMA GIT



VERSIONAMENTO e RAMIFICAÇÃO



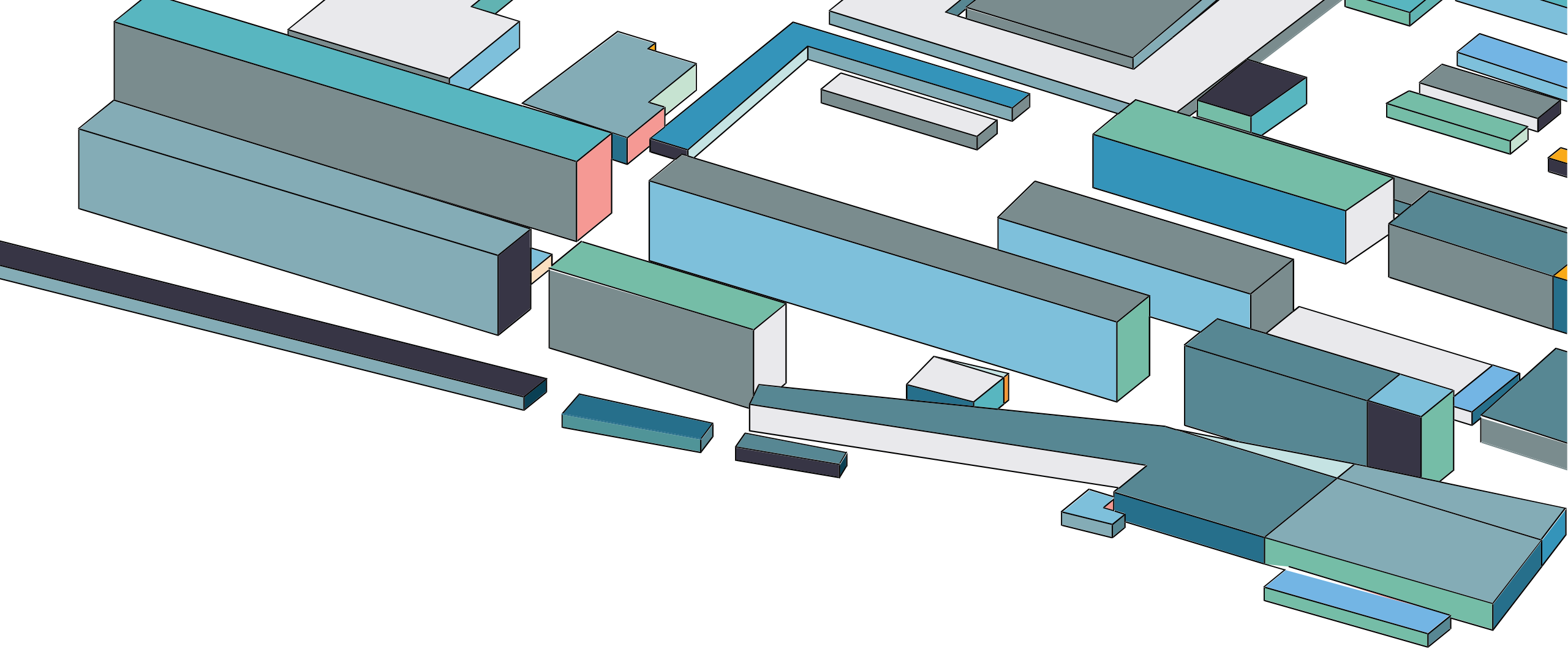
VERSIONAMENTO e RAMIFICAÇÃO



MASTER: MANTER A LINHA DO TEMPO “SAGRADA”

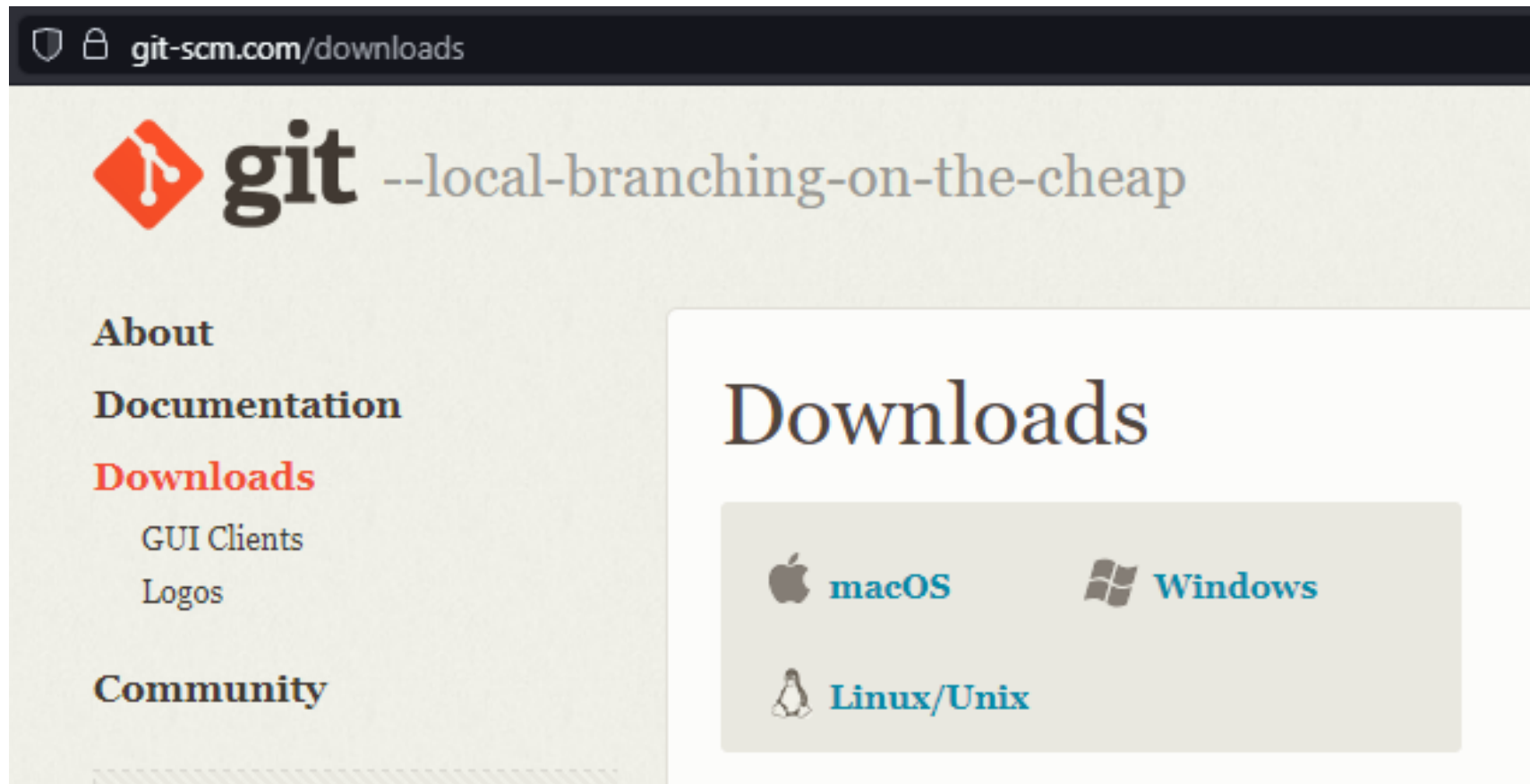


VERSIONAMENTO e RAMIFICAÇÃO



COMEÇANDO NO GIT

COMEÇANDO NO GIT



COMEÇANDO NO GIT



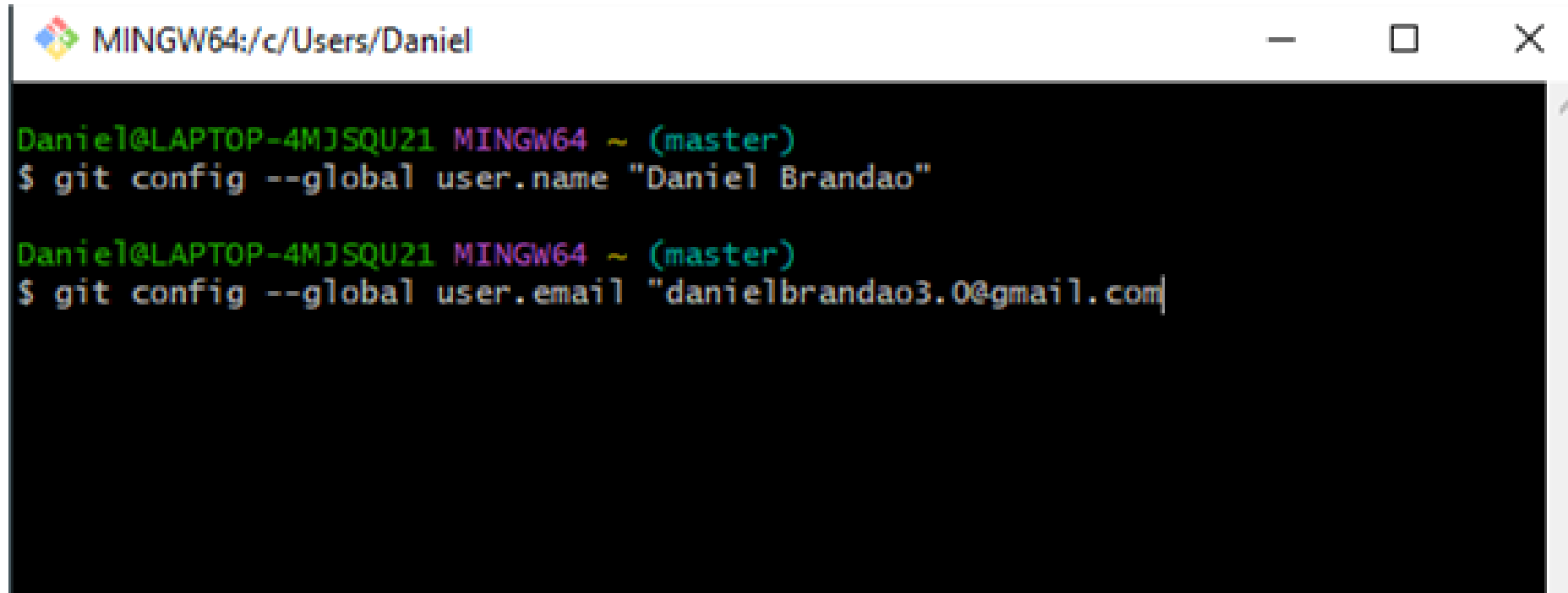
- O Git Bash é o aplicativo para ambientes do Microsoft Windows que oferece a camada de emulação para a experiência de linha de comando Git.
- Bash é acrônimo para "Bourne Again Shell". Shells são aplicativos terminais usados como interface em sistemas operacionais por meio de comandos gravados.

COMEÇANDO NO GIT



```
MINGW64:/c/Users/Daniel  
Daniel@LAPTOP-4MJSQU21 MINGW64 ~ (master)  
$
```

CONFIGURANDO O GIT



A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/Daniel". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal shows two commands being executed to configure Git globally. The first command sets the user name to "Daniel Brandao", and the second command sets the user email to "danielbrandao3.0@gmail.com". The prompt shows the user is in the "master" branch of a repository.

```
Daniel@LAPTOP-4MJSQU21 MINGW64 ~ (master)
$ git config --global user.name "Daniel Brandao"

Daniel@LAPTOP-4MJSQU21 MINGW64 ~ (master)
$ git config --global user.email "danielbrandao3.0@gmail.com"
```

VENDO AS CONFIGURAÇÕES

```
Daniel@LAPTOP-4MJSQU21 MINGW64 ~ (master)
$ git config user.name
Daniel Brandao
```

```
Daniel@LAPTOP-4MJSQU21 MINGW64 ~ (master)
$ git config user.email
danielbrandao3.0@gmail.com
```

CRIANDO PASTA E ACESSANDO

```
Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents  
$ mkdir pos-web
```

```
Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents  
$ cd po  
POTE DA GRATIDAO.docx pos-web/
```

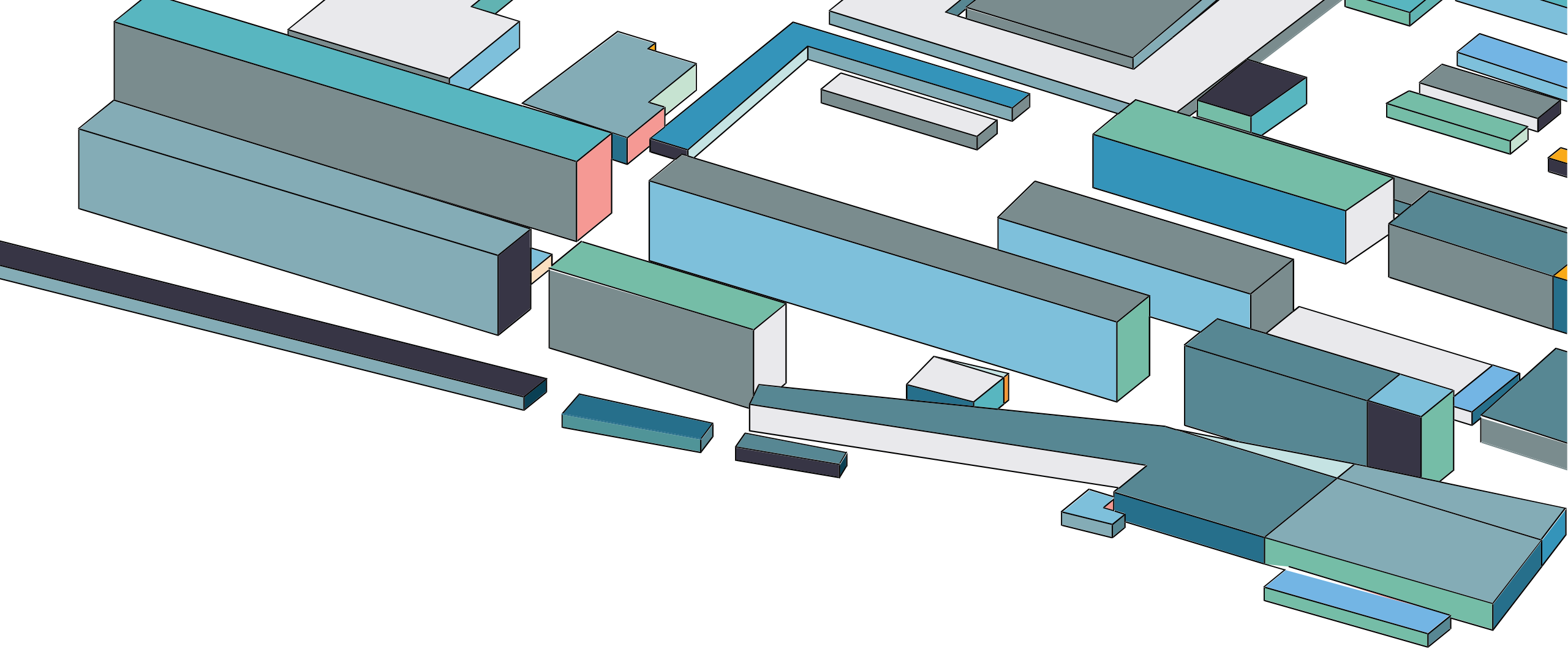
```
Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents  
$ cd pos-web/
```

INICIANDO REPOSITÓRIO LOCAL

```
Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents/pos-web
$ git init
Initialized empty Git repository in E:/Users/Daniel/Documents/pos-web/.git/

Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents/pos-web (master)
$ cd .git

Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents/pos-web/.git (GIT_DIR!)
$ ls
HEAD  config  description  hooks/  info/  objects/  refs/
```

PRIMEIROS COMANDOS

PRIMEIROS COMANDOS

Git status – Lista o status atual dos arquivos no repositório atual

```
Daniel@LAPTOP-4MJSQU21 MINGW64 /e/Users/Daniel/Documents/pos-web (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

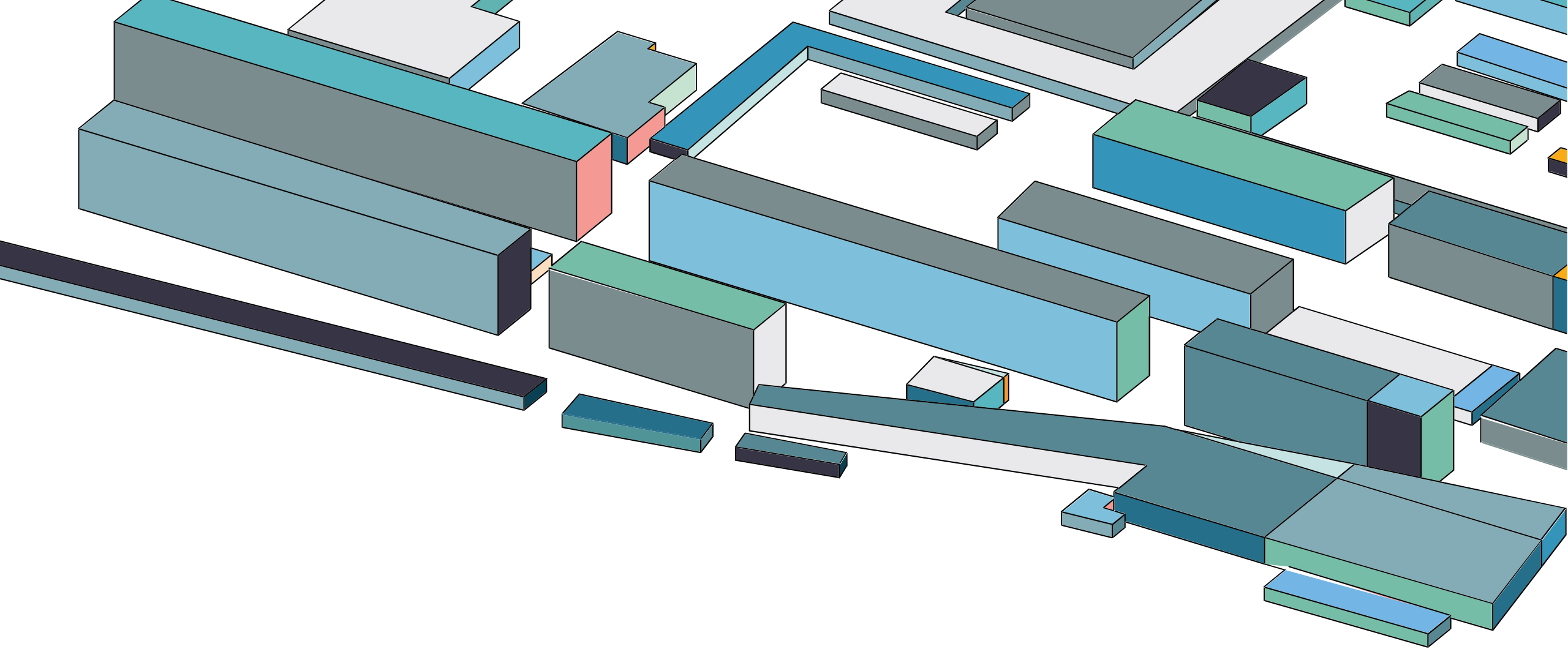
PRIMEIROS COMANDOS

Git add arquivo – adiciona um ou mais arquivos ao status de pronto para dar **commit***

*No contexto de gerenciamento de dados e controle de versão, commit refere-se ao processo de tornar permanente um conjunto de alterações, ou seja, de efetivar as alterações.

PRIMEIROS COMANDOS

Git commit -m “arquivos adicionados” – limpa a lista de arquivos alterados e os prepara para subir ao repositório remoto



CRIANDO REPOSITÓRIO REMOTO



Register for GitHub Universe

Get early bird passes for 20% off

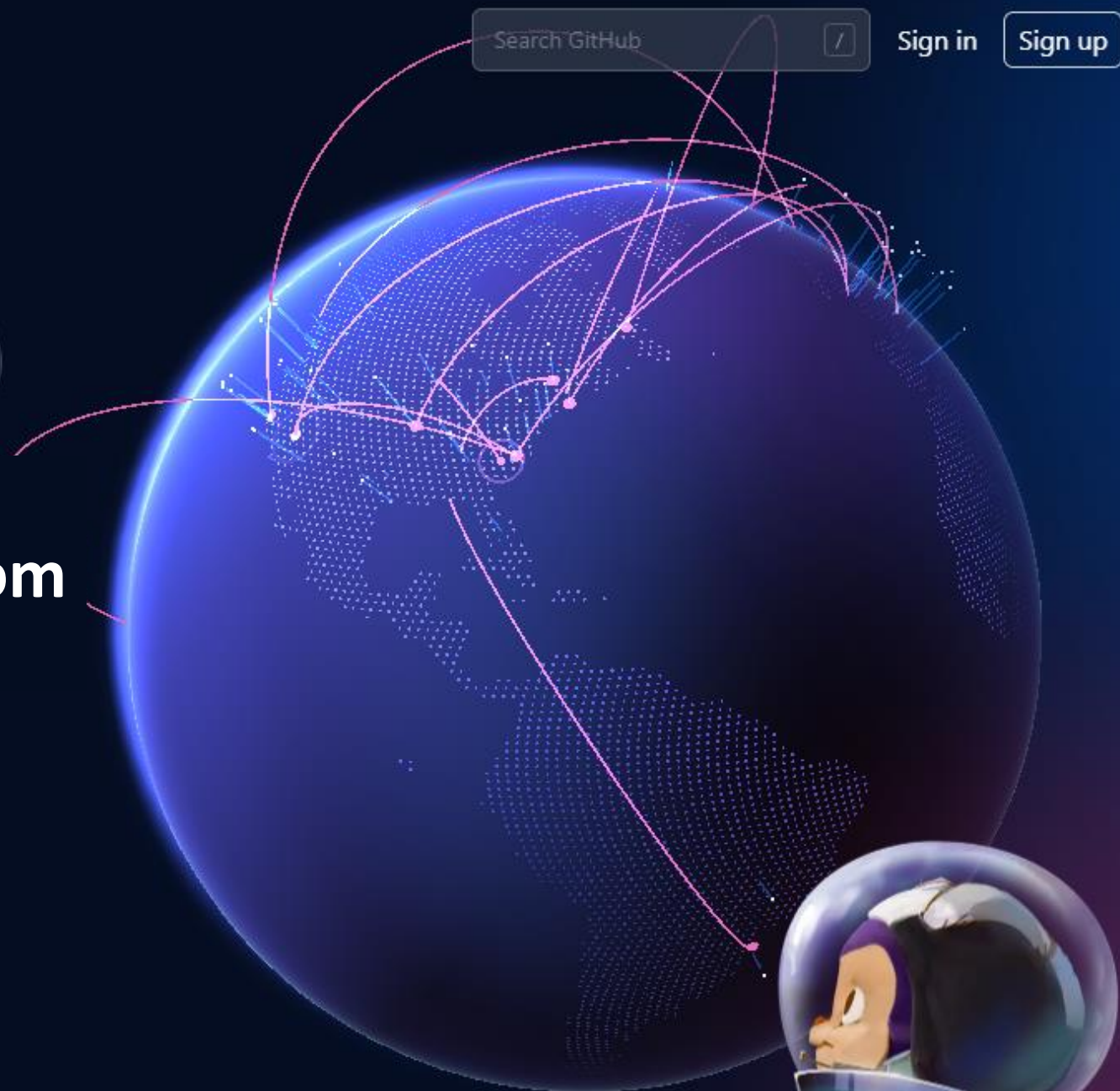


Criar ou logar na conta Github.com

The complete developer platform to build,
scale, and deliver secure software.

Email address

Sign up for GitHub



CRIANDO REPOSITÓRIO REMOTO



Search or jump to...



[Pull requests](#)

[Issues](#)

[Marketplace](#)

[Explore](#)



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Owner *

Repository name *



danielbrandao ▾



Great repository names are short and memorable. Need inspiration? How about [curly-computing-machine?](#)

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)

CONECTANDO AO REPOSITÓRIO REMOTO

Git remote add origin <http://github.com/user/001.git> –
conectando ao repositório remoto do endereço citado

CONECTANDO AO REPOSITÓRIO REMOTO

git init

git add README.md

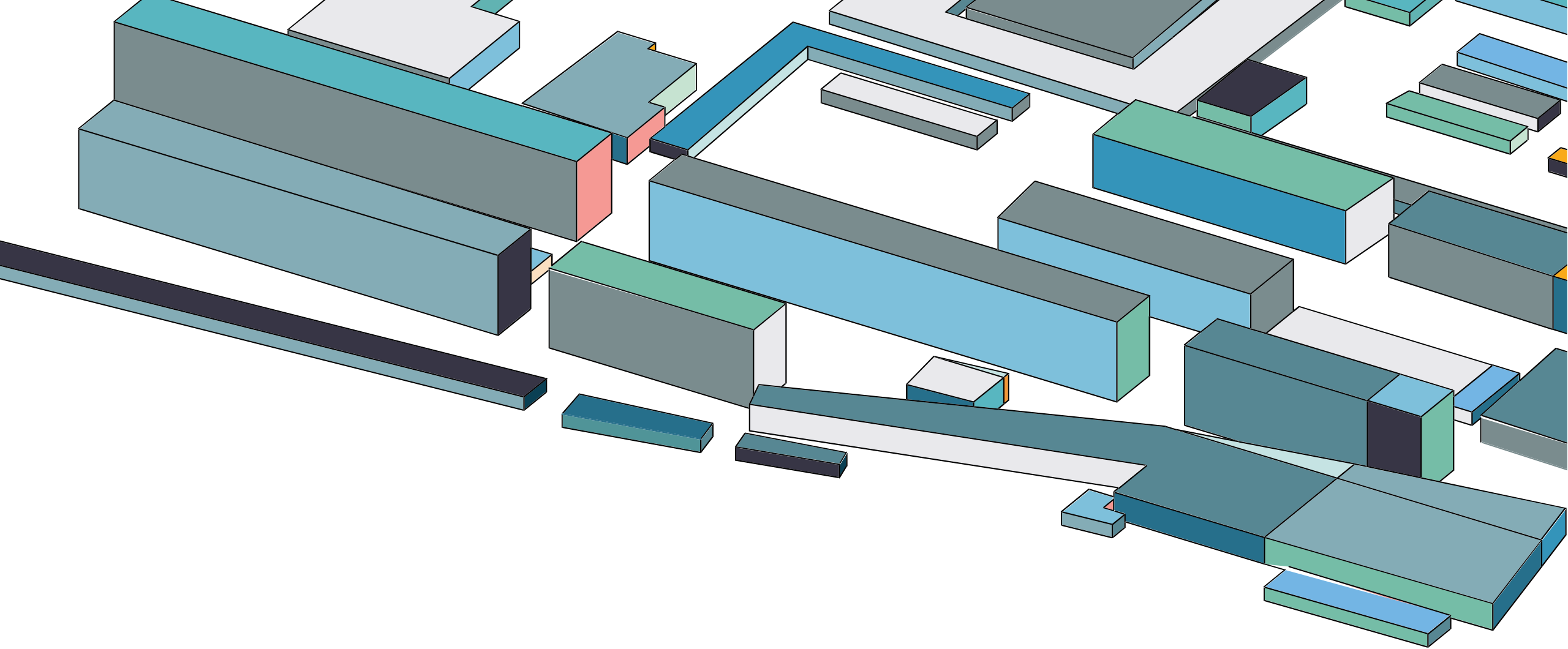
git commit -m "first commit"

git branch -M main

git remote add origin

<https://github.com/danielbrandao/001.git>

git push -u origin main



PROJETO PRÁTICO

PROJETO PRÁTICO

Desenvolvimento Web

PASSO 1

Criaremos uma página web para poder subir ao repositório.

PASSO 2

Criaremos o repositório local e o remoto, conectando um ao outro.

PASSO 3

Versionaremos o repositório e submeteremos os arquivos ao GitHub.

O QUE O PROJETO PRECISA TER



#1

INDEX HTML COMO
PÁGINA INICIAL



#2

Outras Páginas contendo um
conteúdo como QUEM SOMOS
e CONTATOS



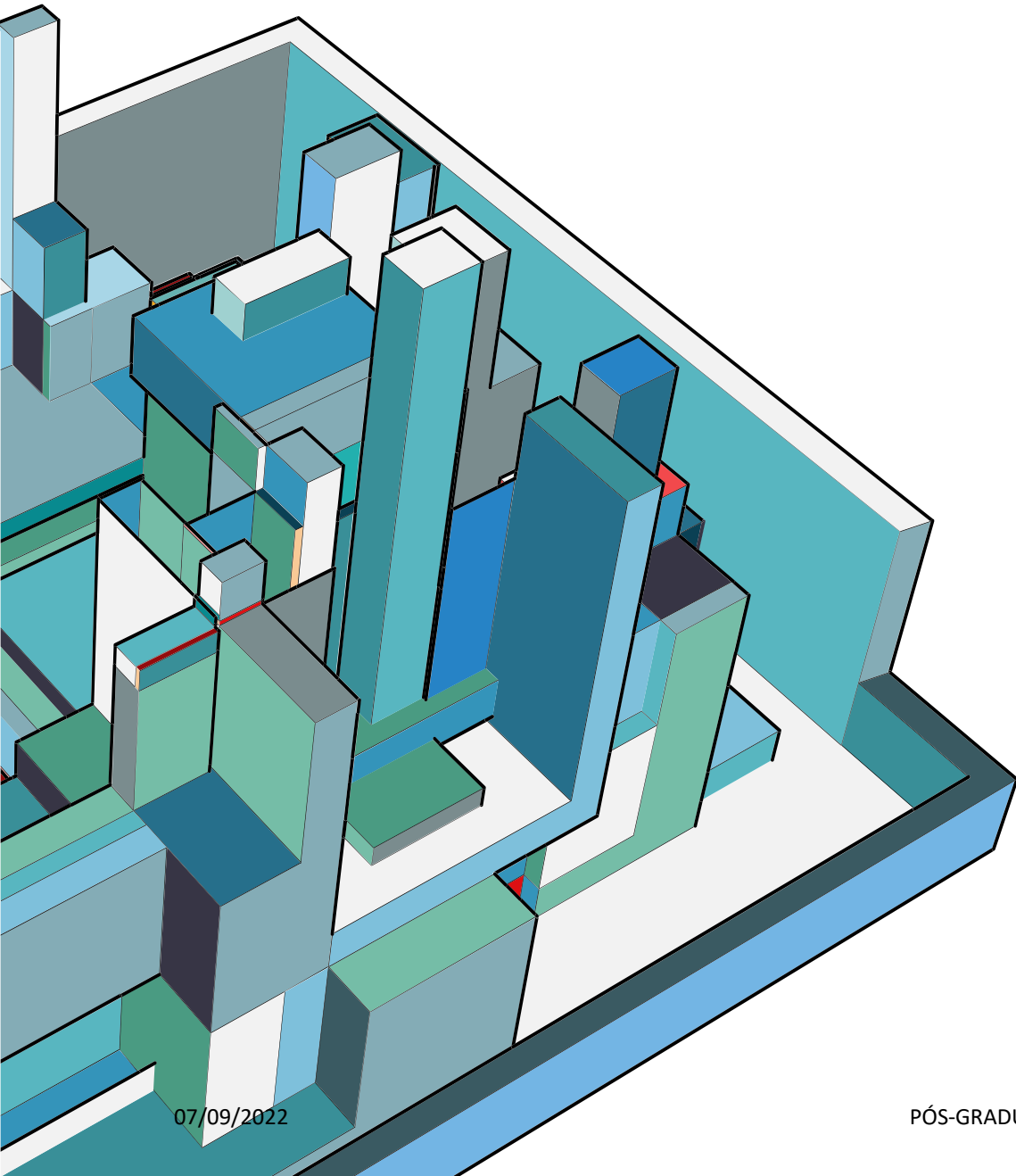
#3

Suas informações devem estar
no projeto, que deve ser
entregue como forma de
pontuação.

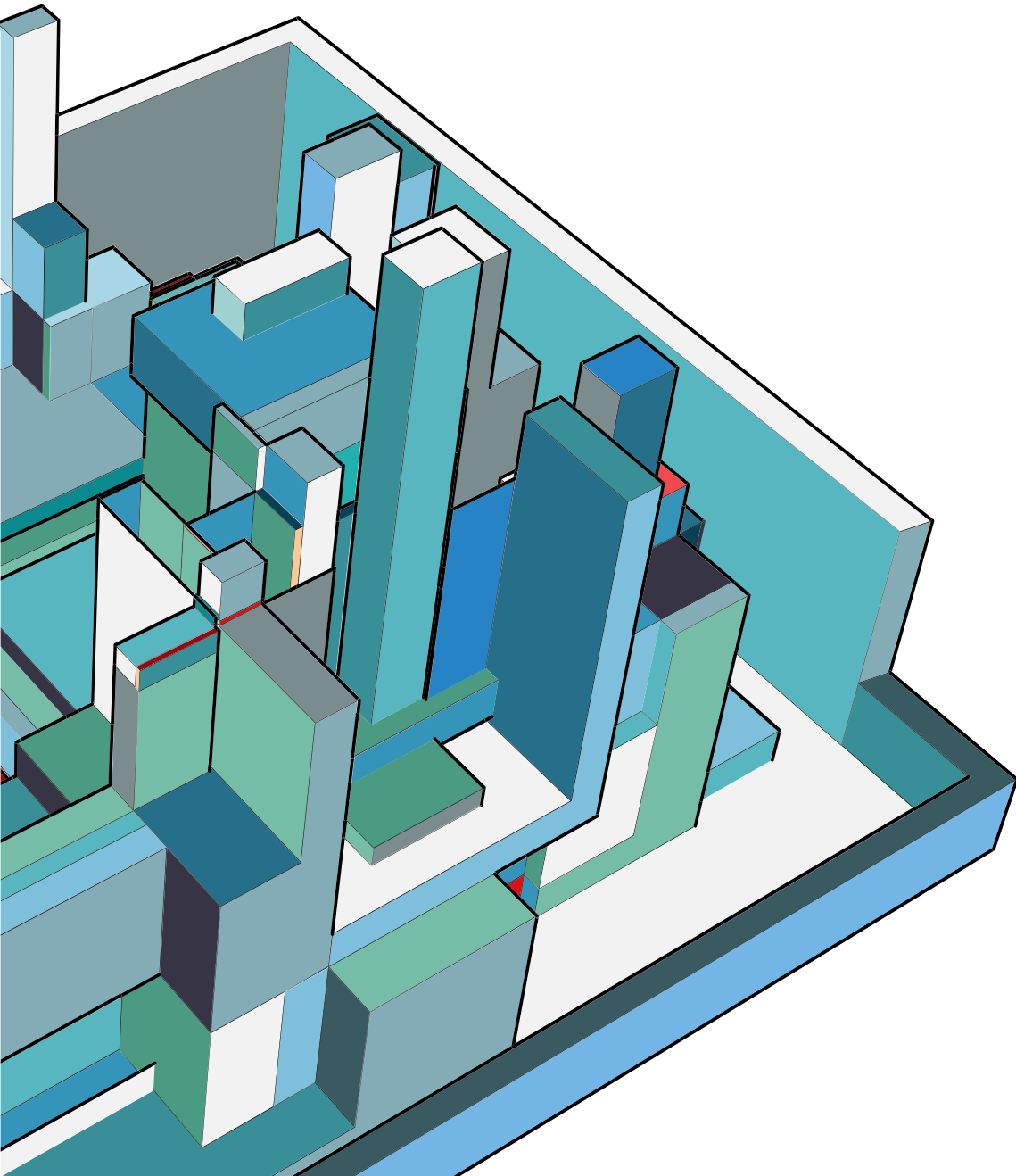
AO FINAL

- Entrega do link do repositório para o e-mail:

professordanielbrandao@gmail.com



**ALGUMA
DÚVIDA?**



Arquivos do Módulo

<https://bit.ly/posweb2022>



OBRIGADO

Prof. Daniel Brandão

(83) 98896-2125

professordanielbrandao@Gmail.com

@ProfDanielBrandao