

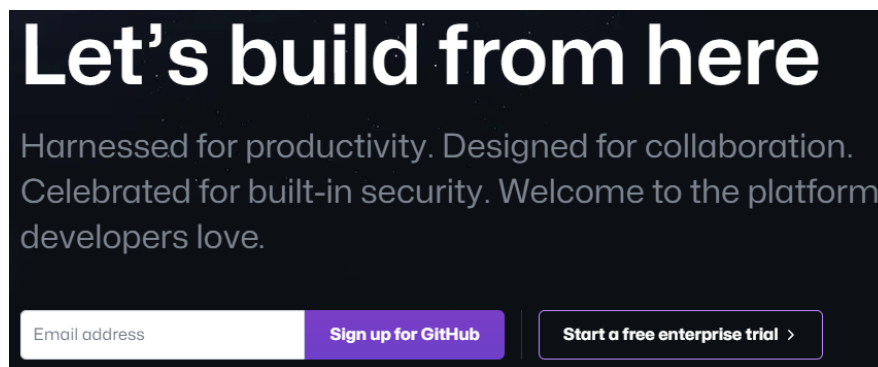
# Programação de Computadores II

## Desenvolvimento Web

### git e GitHub: Dia 1 e 2

**GitHub:** é uma plataforma de hospedagem de código-fonte e colaboração. Ela utiliza o sistema de controle de versão *git* para permitir que os desenvolvedores armazenem, compartilhem e trabalhem em projetos de forma colaborativa. Além disso, o *GitHub* oferece recursos como controle de versão, gerenciamento de problemas, integração contínua e hospedagem de páginas da web, tornando-se uma plataforma popular para desenvolvimento de software.

- Para utilizar o *GitHub* acesse o site <https://github.com/> e crie uma conta sua nele usando o e-mail de sua conta Google.





**git:** é um sistema de controle de versão distribuído amplamente utilizado no desenvolvimento de software. Ele permite rastrear e gerenciar alterações em arquivos, facilitando o trabalho colaborativo, o controle de histórico e a criação de ramificações independentes para desenvolvimento paralelo. Você pode instalar e usar localmente no seu computador. Ele permite que você controle as alterações em seus arquivos e mantenha um histórico de versões.

- Baixe o *git* no site <https://git-scm.com/> e instale-o em seu computador.



## Repositórios:

Um *repositório* (em inglês *repository*) do *GitHub* é uma pasta que armazena e gerencia projetos. Ele permite o controle de versão, gerenciamento de problemas e colaboração. Desenvolvedores podem clonar repositórios, fazer alterações e enviá-las de volta, mantendo um histórico completo das modificações. Os repositórios podem ser *públicos* ou *privados*.


Para criar um repositório no seu usuário do GitHub, acesse sua conta e clique no botão  , que fica na parte superior esquerda do navegador, ou, escolha a primeira opção do botão  , que fica na parte superior direita.

- ✓ Configure-o conforme mostrado nos quadros de cor vermelha na tela a seguir:

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)



Required fields are marked with an asterisk (\*).

Owner *	Repository name *
 Lavestein ▾	/ PCII
✓ pcii is available.	

Great repository names are short and memorable. Need inspiration? How about [congenial-succotash](#) ?

Description (optional)

Treinamento prático para as aulas de PCII

<input checked="" type="radio"/>  Public
Anyone on the internet can see this repository. You choose who can commit.
<input type="radio"/>  Private
You choose who can see and commit to this repository.

Initialize this repository with:

<input checked="" type="checkbox"/> Add a README file
This is where you can write a long description for your project. <a href="#">Learn more about READMEs.</a>

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

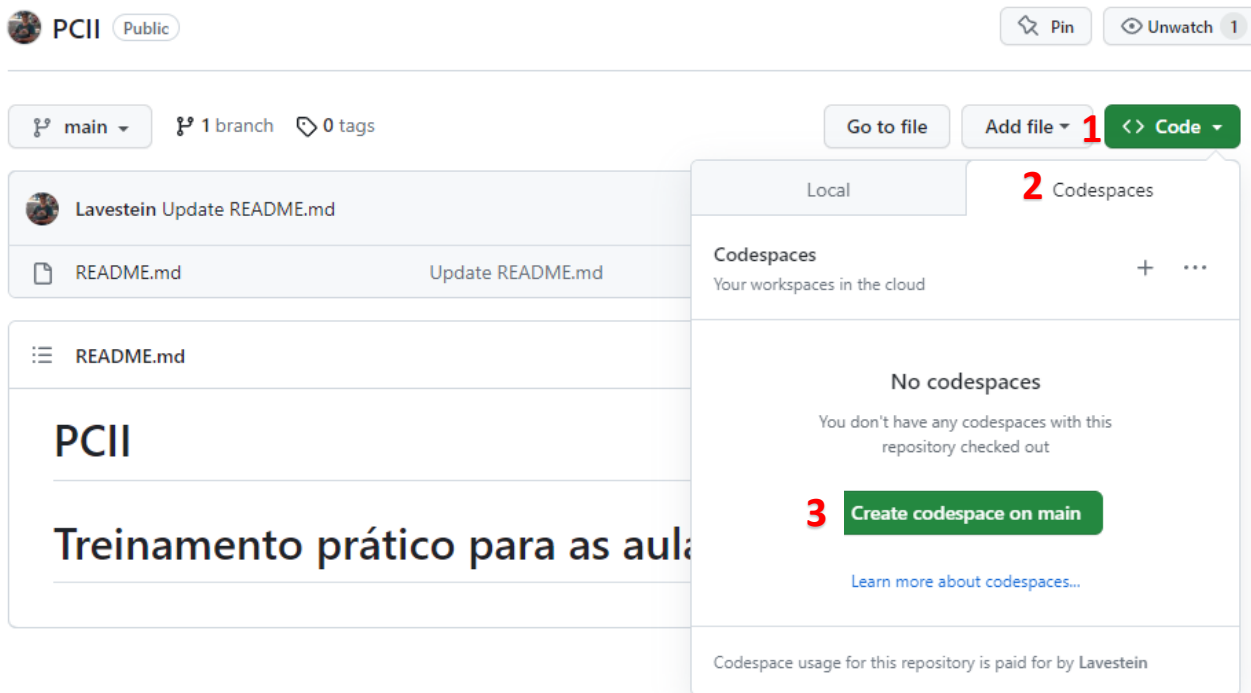
Clique aqui

Create repository

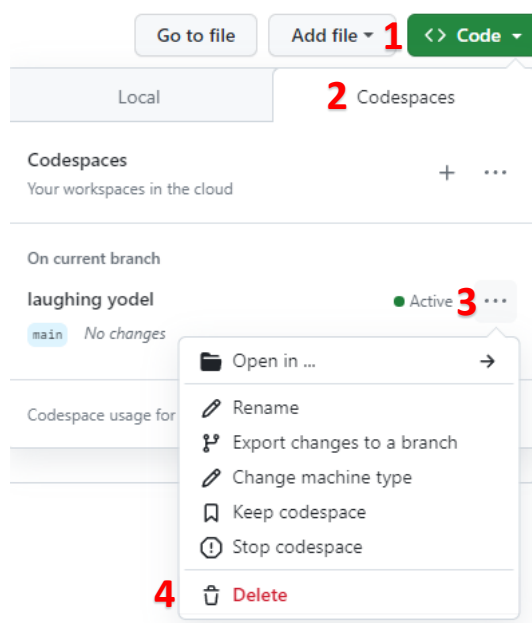
## Ambiente de Desenvolvimento:

**Codespaces:** é um ambiente de desenvolvimento online disponibilizado pelo *GitHub* pronto para uso. Ele elimina a necessidade de configurar um ambiente local, fornecendo todas as dependências, configurações e extensões necessárias. Os *Codespaces* permitem a edição, compilação e teste de código diretamente no navegador, proporcionando uma experiência integrada. Eles são especialmente úteis para projetos complexos, facilitando a colaboração e permitindo uma rápida contribuição, independentemente das configurações locais. O ambiente fornecerá um editor de código (*VSCode*) e um terminal que por padrão roda Linux.

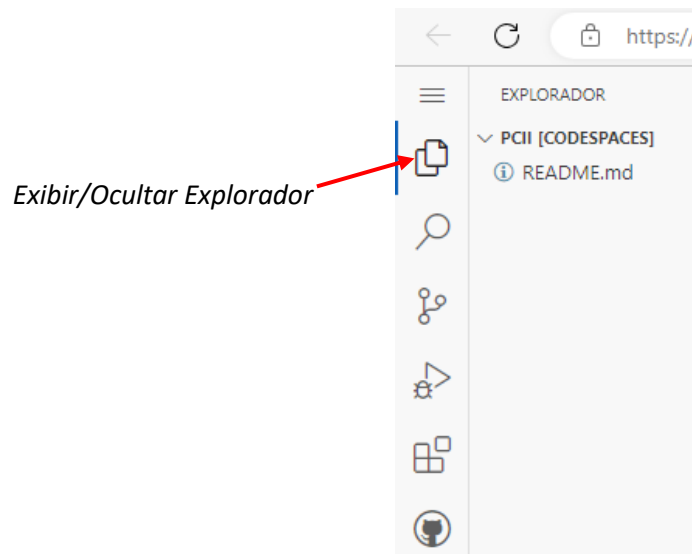
- ✓ Para criar um *Codespaces* em um repositório do *GitHub* clique nos passos **1**, **2** e **3** a seguir:



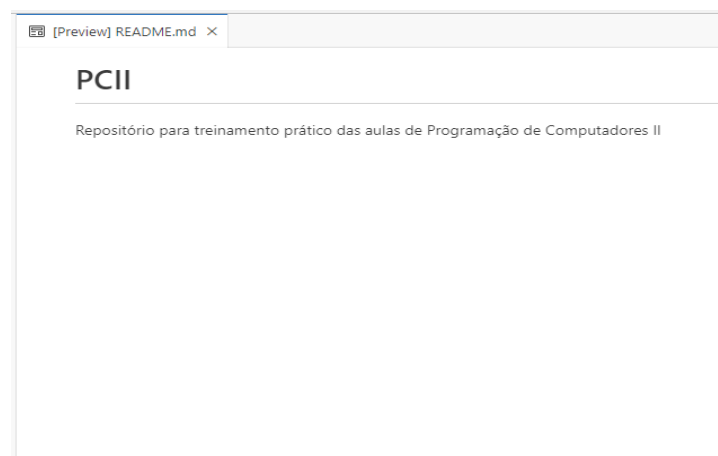
- ✓ Para apagar um *Codespaces* em um repositório do *GitHub* clique nos passos **1**, **2**, **3** e **4** a seguir:



**Partes da tela do Codespaces:** observe as partes do *Codespace* nas figuras a seguir:



**Figura 1** Explorador (árvores de arquivos)



**Figura 2** Editores Abertos

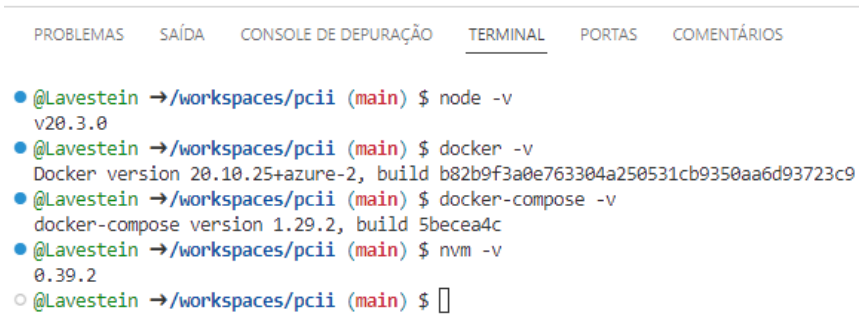


**Figura 3** Terminal Linux

### Vamos praticar um pouco:

1. Esconda o explorador
2. Digite os seguintes comandos no Terminal Linux do Codespace:
  - ✓ **node -v**: exibir a versão do Node.js (ambiente de execução de JavaScript do lado do servidor)
  - ✓ **docker -v**: exibir a versão do docker (virtualizador de contêineres (Isolamento de aplicativos))
  - ✓ **docker-compose -v**: exibir a versão do docker-compose (gerenciador de contêineres )
  - ✓ **nvm -v**: exibir a versão do gerenciador de versões do Node.js

O resultado deverá ficar parecido com o da tela a seguir:




```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein →/workspaces/pcii (main) $ node -v
v20.3.0
● @Lavestein →/workspaces/pcii (main) $ docker -v
Docker version 20.10.25+azure-2, build b82b9f3a0e763304a250531cb9350aa6d93723c9
● @Lavestein →/workspaces/pcii (main) $ docker-compose -v
docker-compose version 1.29.2, build 5becea4c
● @Lavestein →/workspaces/pcii (main) $ nvm -v
0.39.2
○ @Lavestein →/workspaces/pcii (main) $
```

**Obs.:** Após a execução em seu terminal aparecerá o nome do seu usuário após o sinal de @

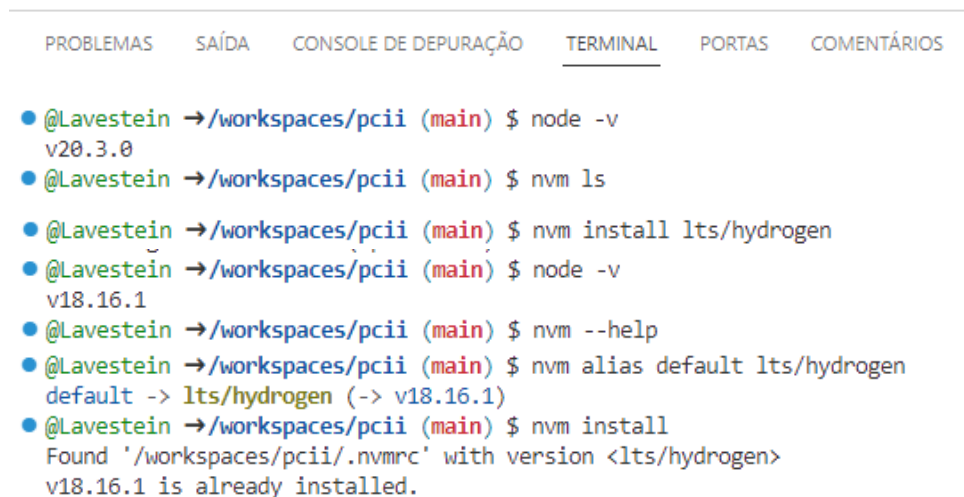
**Dicas** - experimente executar os seguintes comandos:

- ✓ Digite o comando **clear** ou tecle **CTRL + L** para limpar a tela do terminal.
- ✓ Tecle **CTRL + '**  para exibir/ocultar a tela do terminal.
- ✓ Configure entre outras coisas, o tema do Codespace através do botão  que se encontra na parte inferior esquerda da sua tela.

### Definindo a Fundação do nosso projeto (Node.js): Dia 3

Digite os seguintes comandos no Terminal Linux do Codespace:

- ✓ **node -v**: exibir a versão do Node.js (ambiente de execução de JavaScript do lado do servidor)
- ✓ **nvm ls**: lista as versões atuais disponíveis do Node.js. As LTS são Long Term Support (Suporte de Longo Prazo).
- ✓ **nvm install lts/hydrogen**: baixa, instala e configura a versão lts/hydrogen na instância do terminal.
- ✓ **node -v**
- ✓ **nvm --help**: documentação do NVM (Node Version Manager)
- ✓ **nvm alias default lts/hydrogen**: faz com que o padrão do nosso terminal aponte para a versão *lts/hydrogen*. Mesmo se fechar o terminal local a configuração permanecerá a *lts/hydrogen*.
- ✓ Use o **Explorador** e crie um arquivo chamado **.nvmrc** na raiz (/) do projeto. Digite a primeira linha com o texto igual a *lts/hydrogen*, e a segunda linha deixe-a **vazia**. Salve o arquivo com **CTRL + S**.
- ✓ **nvm install**: instalará a versão contida no arquivo .nvmrc



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein →/workspaces/pcii (main) $ node -v
v20.3.0
● @Lavestein →/workspaces/pcii (main) $ nvm ls

● @Lavestein →/workspaces/pcii (main) $ nvm install lts/hydrogen
● @Lavestein →/workspaces/pcii (main) $ node -v
v18.16.1
● @Lavestein →/workspaces/pcii (main) $ nvm --help
● @Lavestein →/workspaces/pcii (main) $ nvm alias default lts/hydrogen
default -> lts/hydrogen (-> v18.16.1)
● @Lavestein →/workspaces/pcii (main) $ nvm install
Found '/workspaces/pcii/.nvmrc' with version <lts/hydrogen>
v18.16.1 is already installed.
```

Comandos digitados no **Terminal Linux**

#### Conceitos Importantes:

**Node.js**: Ambiente de execução JavaScript no lado do servidor.

**Next.js**: Framework React para desenvolvimento web com recursos avançados como renderização do lado do servidor (SSR) e geração de páginas estáticas.

**React**: Biblioteca JavaScript para construção de interfaces de usuário interativas e reutilizáveis.

**React DOM**: Biblioteca para renderização de componentes React no DOM do navegador.

**DOM**: estrutura que representa a estrutura de uma página web e permite a manipulação dos elementos dessa página através de código.

Criação do arquivo **Manifesto** (*package.json*): lista as dependências do projeto, como versões do NextJS, React, etc. Para isso, digite o seguinte comando no Terminal Linux do Codespace:

- ✓ **npm init**: cria e configura o arquivo package.json

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein →/workspaces/pcii (main) $ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (pcii)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC) MIT
About to write to /workspaces/pcii/package.json:

{
  "name": "pcii",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "MIT"
}

Is this OK? (yes)
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.8.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.8.0
npm notice Run npm install -g npm@9.8.0 to update!
npm notice
○ @Lavestein →/workspaces/pcii (main) $ 

```

Resultado dos comandos no **Terminal Linux**

**Continuação:**

- ✓ **npm install [next@13.1.6](#):** instala o Next.Js na versão 13.1.6.
- ✓ **npx next -v:** exibe a versão do Next.JS.
- ✓ **npm install [react@18.2.0](#):** instala o React na versão 18.2.0.
- ✓ **npm install [react-dom@18.2.0](#):** instala o React-dom na versão 18.2.0.

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein →/workspaces/pcii (main) $ npm install next@13.1.6

added 18 packages, and audited 19 packages in 7s

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
● @Lavestein →/workspaces/pcii (main) $ npx next -v
Next.js v13.1.6
● @Lavestein →/workspaces/pcii (main) $ npm install react@18.2.0

up to date, audited 19 packages in 761ms

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
● @Lavestein →/workspaces/pcii (main) $ npm install react-dom@18.2.0

up to date, audited 19 packages in 352ms

3 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
○ @Lavestein →/workspaces/pcii (main) $ █
```

Resultado dos comandos no **Terminal Linux**

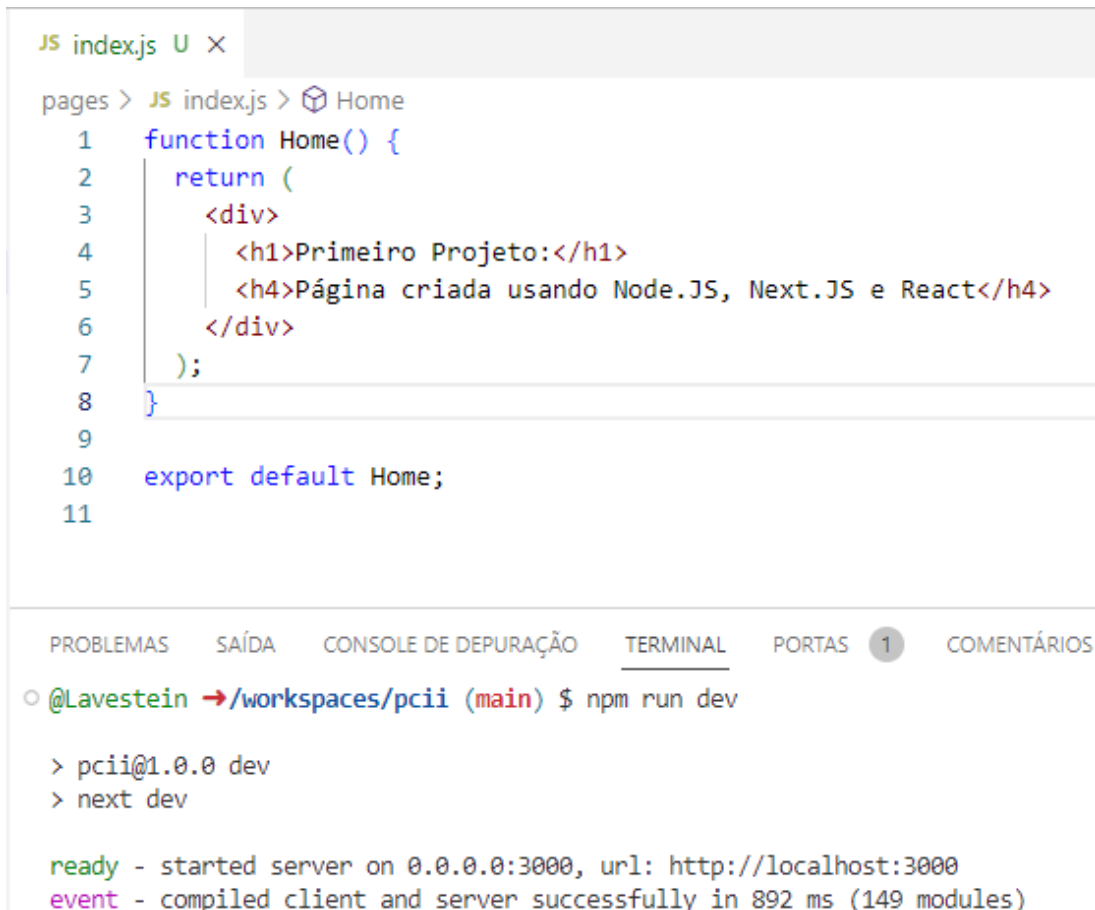


- ✓ Altere o arquivo **package.json**, conforme indicado no quadro vermelho a seguir:

```
{ } package.json U X
{ } package.json > ...
1  {
2    "name": "pcii",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
   ▶ Debug
6    "scripts": {
7      "dev": "next dev"
8    },
9    "author": "",
10   "license": "MIT",
11   "dependencies": {
12     "next": "^13.1.6",
13     "react": "^18.2.0",
14     "react-dom": "^18.2.0"
15   }
16 }
```

### Criando a primeira página WEB:

- ✓ Crie uma pasta chamada **pages** na raiz (/) do seu projeto
- ✓ Crie um arquivo chamado **index.js** dentro da pasta **pages** e digite os seguintes comandos, conforme demonstrado a seguir:
- ✓ Execute o comando **npm run dev** no Terminal Linux



The image shows a VS Code editor window with a file named `index.js` in the `pages` directory. The code defines a `Home` function that returns a JSX element with an `h1` and an `h4`. Below the editor, the terminal shows the command `npm run dev` being executed, which starts the development server on port 3000.

```
JS index.js U X
pages > JS index.js > Home
1 function Home() {
2   return (
3     <div>
4       <h1>Primeiro Projeto:</h1>
5       <h4>Página criada usando Node.JS, Next.JS e React</h4>
6     </div>
7   );
8 }
9
10 export default Home;
11
```

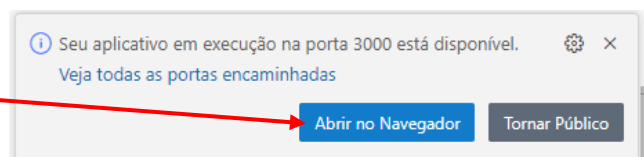
PROBLEMAS SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS 1 COMENTÁRIOS

@Lavestein →/workspaces/pcii (main) \$ npm run dev

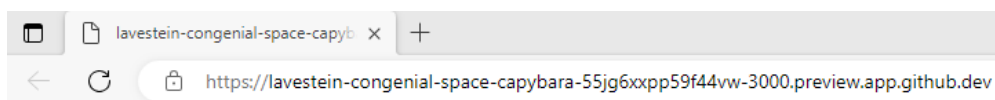
> pcii@1.0.0 dev  
> next dev

ready - started server on 0.0.0.0:3000, url: http://localhost:3000  
event - compiled client and server successfully in 892 ms (149 modules)

- ✓ Clique no botão **Abrir no Navegador**



- ✓ Verifique em seu navegador se este foi o resultado final:





## Primeiro Projeto:

Página criada usando Node.JS, Next.JS e React

- ✓ Parabéns! Você conclui seu primeiro projeto em **Node.JS** com **Next.JS** e **React**.

### Dicas:

- ✓ Execute os procedimentos a seguir para tornar **Pública** a visibilidade de seu endereço local:
- ✓ Clique no botão  1 , clique com o botão direito sobre o campo **Visibilidade** e marque a opção Visibilidade da porta para **Public**.
- ✓ Caso queira visualizar o resultado final, clique em e  escolha uma das opções: **Abrir no Navegador** ou **Visualizar no Editor**
- ✓ Para desativar a execução do servidor volte para o Terminal e tecle **CTRL + C**.

### Comandos Git:

- ✓ Execute o comando `ls -lA` no Terminal Linux: exibe a lista de todos os arquivos do diretório, incluindo os que estão ocultos (.nomearquivo).

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein → /workspaces/pcii (main) $ ls -lA
total 44
drwxrwxrwx+ 9 codespace root      4096 Jul 17 11:28 .git
drwxrwxrwx+ 5 codespace codespace 4096 Jul 17 14:25 .next
-rw-rw-rw-  1 codespace codespace  13 Jul 17 13:54 .nvmrc
-rw-rw-rw-  1 codespace root       6 Jul 17 11:28 README.md
drwxrwxrwx+ 19 codespace codespace 4096 Jul 17 12:19 node_modules
-rw-rw-rw-  1 codespace codespace 13890 Jul 17 12:21 package-lock.json
-rw-rw-rw-  1 codespace codespace  263 Jul 17 13:54 package.json
drwxrwxrwx+ 2 codespace codespace 4096 Jul 17 12:33 pages
○ @Lavestein → /workspaces/pcii (main) $

```

A pasta **.git** armazena informações e histórico de controle de versão do seu projeto. Experimente executar o comando `ls -l .git` para visualizar o conteúdo dela.

Na próxima etapa utilizaremos o comando **git** para executar diversas operações de controle de versão, como gerenciar repositórios, fazer commits, criar branches, mesclar alterações e muito mais.

**Obs.:** Uma branch é uma linha de desenvolvimento separada que permite trabalhar em alterações isoladas no controle de versão Git.

#### Trabalhando com o comando **git**:

- ✓ O comando **git log** exibe o histórico de *commits* do repositório Git.
- ✓ O comando **git log --stat** exibe o histórico de commits do repositório Git com informações estatísticas sobre as alterações feitas em cada commit.
- ✓ O comando **git status** exibe o estado atual do repositório Git, mostrando quais arquivos foram modificados, adicionados, removidos ou estão em espera para serem “*commitados*”.
- ✓ O comando **git log --oneline** exibe o histórico de commits de forma resumida em uma linha por commit.
- ✓ O comando **git diff** mostra as diferenças entre o estado atual dos arquivos no diretório de trabalho e a versão mais recente registrada no repositório Git.
- ✓ O comando **git add** é usado para adicionar arquivos ao índice (staging area), preparando-os para serem incluídos no próximo commit.
- ✓ O comando **git add -A** é usado para adicionar todas as alterações, incluindo arquivos modificados, novos arquivos e arquivos removidos, ao índice para serem incluídos no próximo commit.
- ✓ O comando **git commit** é usado para criar um commit, registrando as alterações preparadas no repositório Git.
- ✓ O comando **git commit -m "mensagem do commit"** é usado para criar um commit com uma mensagem descritiva.
- ✓ O comando **git branch** lista todas as branches no repositório local.
- ✓ O comando **git push** envia os commits locais para um repositório remoto.
- ✓ O comando **git pull** obtém as alterações mais recentes de um repositório remoto e mescla com o branch atual.
- ✓ O comando **git push --force** força o envio dos commits locais para um repositório remoto, substituindo o histórico existente.

#### Estágios dos Commits:

- 1) **Untracked files:** Arquivos que não estão sendo rastreados pelo Git.
- 2) **Modified:** Arquivos que foram modificados no diretório de trabalho.
- 3) **Staged:** Arquivos que foram preparados para serem incluídos no próximo commit.
- 4) **Committed:** Alterações confirmadas e registradas permanentemente no histórico do Git.

- ✓ Execute os comandos da tela a seguir no Terminal Linux:

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS  COMENTÁRIOS

● @Lavestein →/workspaces/pcii (main) $ git log
commit c854d81ea833b0419baf1c21570e129089c0449a (HEAD -> main)
Author: Ronaldo Lavestein <71901765+Lavestein@users.noreply.github.com>
Date:   Mon Jul 17 08:28:04 2023 -0300

    Initial commit

● @Lavestein →/workspaces/pcii (main) $ git log --stat
commit c854d81ea833b0419baf1c21570e129089c0449a (HEAD -> main)
Author: Ronaldo Lavestein <71901765+Lavestein@users.noreply.github.com>
Date:   Mon Jul 17 08:28:04 2023 -0300

    Initial commit

    README.md | 1 +
    1 file changed, 1 insertion(+)

● @Lavestein →/workspaces/pcii (main) $ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

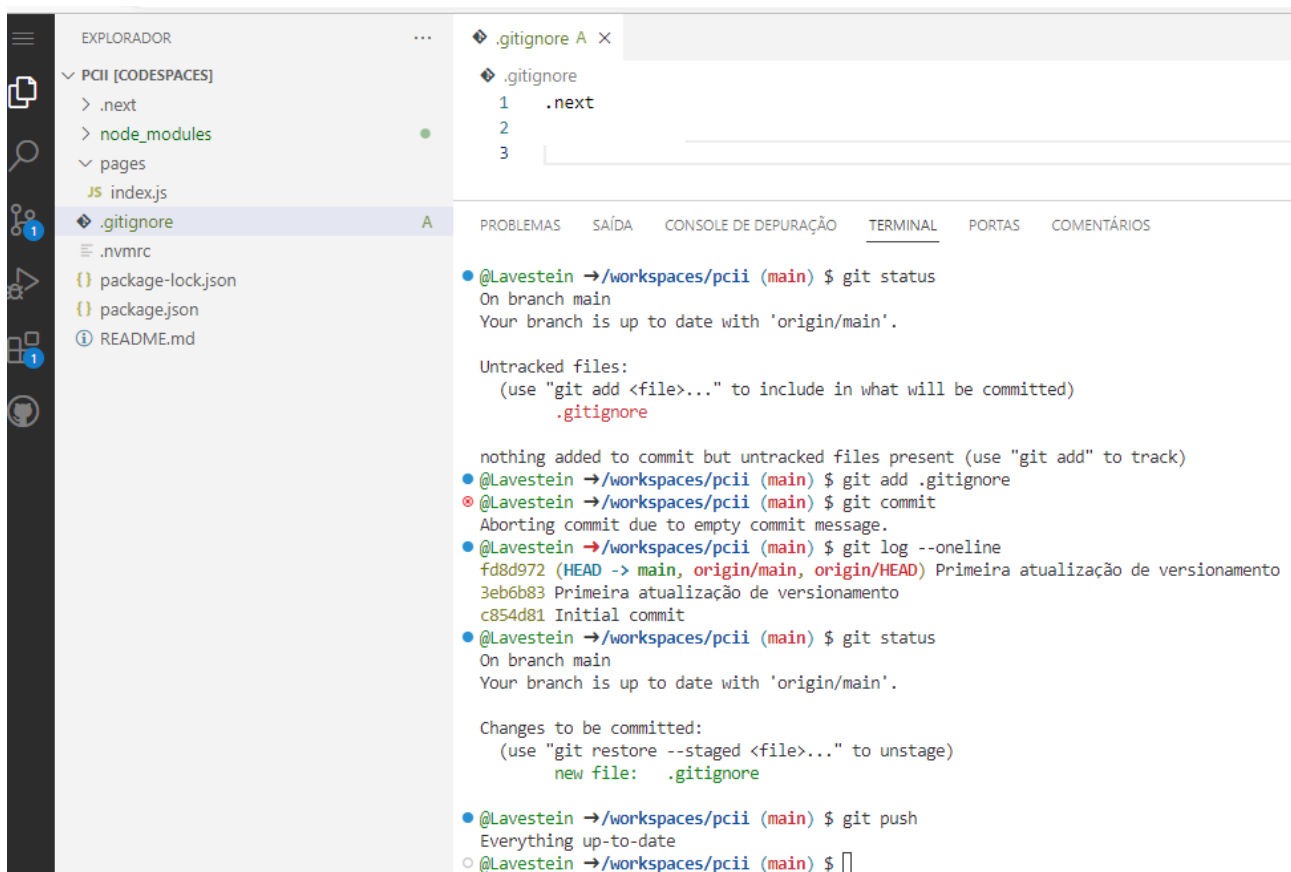
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .next/
    .nvmrc
    node_modules/
    package-lock.json
    package.json
    pages/

nothing added to commit but untracked files present (use "git add" to track)
● @Lavestein →/workspaces/pcii (main) $ git add -A
● @Lavestein →/workspaces/pcii (main) $ git commit -m "Primeira atualização de versionamento"
● @Lavestein →/workspaces/pcii (main) $ git push

```

## Arquivo .gitignore

- ✓ **.gitignore** é um arquivo de configuração usado pelo Git para especificar quais arquivos e diretórios devem ser ignorados e não rastreados pelo controle de versão.
- ✓ Ao adicionar arquivos ou padrões de arquivos no arquivo .gitignore, você informa ao Git quais arquivos ele não deve incluir em commits ou considerar ao verificar o status do repositório. Isso é útil para evitar a inclusão acidental de arquivos sensíveis, como senhas, chaves de API ou arquivos gerados automaticamente, além de evitar o rastreamento de arquivos desnecessários, como arquivos de dependências ou arquivos temporários.
- ✓ O arquivo .gitignore é colocado na raiz do repositório Git e você pode adicionar padrões de arquivos, diretórios ou expressões regulares para especificar quais arquivos devem ser ignorados. O Git, então, não rastreará ou exibirá esses arquivos em operações como git status ou git add.
- ✓ Em resumo, o arquivo .gitignore permite que você especifique quais arquivos ou tipos de arquivos devem ser ignorados pelo Git, evitando que sejam acidentalmente incluídos no controle de versão.
- ✓ Crie o arquivo .gitignore e execute os comandos da tela a seguir no Terminal Linux



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows the file structure of a project named 'PCII [CODESPACES]'. The files listed are: .next, node\_modules, pages, index.js, .gitignore (selected), .nvmrc, package-lock.json, package.json, and README.md. The main editor area shows the content of the .gitignore file, which contains the following text:

```
.next
/
node_modules
/
pages
/
index.js
/
```

Below the editor, the TERMINAL tab is active, displaying the output of several Git commands executed in a Linux terminal:

```
@Lavestein →/workspaces/pcii (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  .gitignore

nothing added to commit but untracked files present (use "git add" to track)
@Lavestein →/workspaces/pcii (main) $ git add .gitignore
@Lavestein →/workspaces/pcii (main) $ git commit
Aborting commit due to empty commit message.
@Lavestein →/workspaces/pcii (main) $ git log --oneline
fd8d972 (HEAD -> main, origin/main, origin/HEAD) Primeira atualização de versionamento
3eb6b83 Primeira atualização de versionamento
c854d81 Initial commit
@Lavestein →/workspaces/pcii (main) $ git status
On branch main
Your branch is up to date with 'origin/main'.

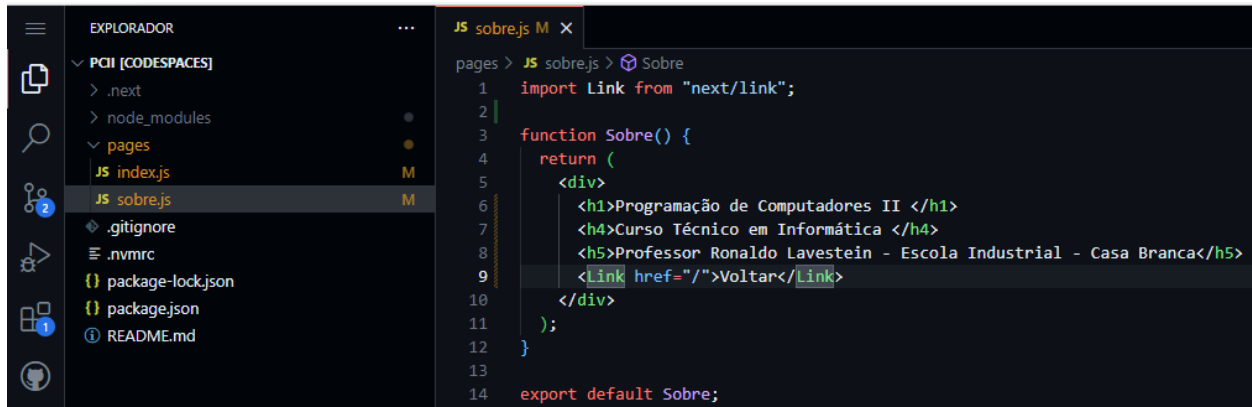
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   .gitignore
@Lavestein →/workspaces/pcii (main) $ git push
Everything up-to-date
@Lavestein →/workspaces/pcii (main) $
```

Criando uma conta **no vercel.com** para hospedar nosso site de maneira gratuita:

- ✓ Acesse o site **vercel.com**
- ✓ Clique no botão **Sign up** para fazer a sua inscrição
- ✓ Marque a opção **Hobby** e logo em seguida digite o seu nome na caixa de texto que irá aparecer.
- ✓ Clique no botão **Continue**
- ✓ Clique no botão **Continue with GitHub** para associar sua conta do GitHub à do Vercel.
- ✓ Abra a tela pop-up que foi aberta e clique no botão **Authorize Vercel** nela.
- ✓ Na seção **Import Git Repository** clique na seta para baixo e escolha **Add GitHub Account**
- ✓ Selecione a pop-up do Vercel, marque **All Repositories** e clique em **Install**
- ✓ Clique no botão Import do repositório desejado e clique no botão **Deploy**
- ✓ Clique no **thumbnail** (tela final do deploy) que apareceu para exibir o site do seu projeto que neste momento já está público na Internet.

**Links no projeto:** Crie outras páginas e acesse-as através do conceito de links.

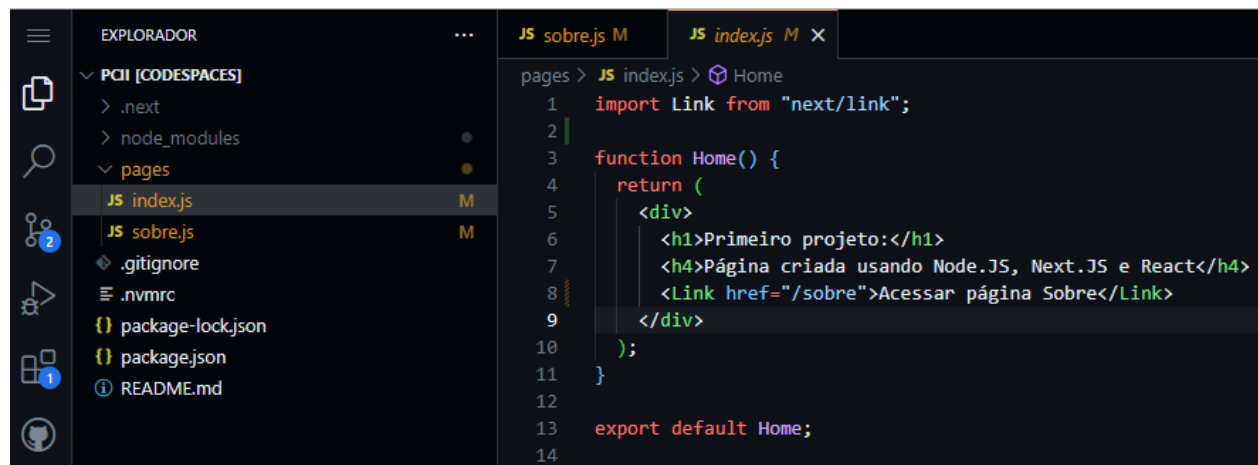
- ✓ Crie um arquivo dentro da página **pages** em seu projeto com o nome de **sobre.js** e deixe-o como exibido na tela a seguir:



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure: PCI [CODESPACES] > .next > node\_modules > pages > JS sobre.js (selected). The main editor area shows the code for **sobre.js**:

```
1 import Link from "next/link";
2
3 function Sobre() {
4   return (
5     <div>
6       <h1>Programação de Computadores II </h1>
7       <h4>Curso Técnico em Informática </h4>
8       <h5>Professor Ronaldo Lavestein - Escola Industrial - Casa Branca</h5>
9       <Link href="/">Voltar</Link>
10    </div>
11  );
12 }
13
14 export default Sobre;
```

- ✓ Depois de salvar o seu arquivo (CTRL + S), altere a página index.js deixando-a conforme a tela a seguir:



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the project structure: PCI [CODESPACES] > .next > node\_modules > pages > JS index.js (selected). The main editor area shows the code for **index.js**:

```
1 import Link from "next/link";
2
3 function Home() {
4   return (
5     <div>
6       <h1>Primeiro projeto:</h1>
7       <h4>Página criada usando Node.JS, Next.JS e React</h4>
8       <Link href="/sobre">Acessar página Sobre</Link>
9     </div>
10  );
11 }
12
13 export default Home;
14
```

- ✓ Teste o projeto no navegador ou visualize-o no editor.

Bons estudos!