



TechTalk Pesquisa

☰ Description	Pesquisa de informação para o TechTalk em Out
📅 Timeline	@August 10, 2023 → October 23, 2023
⚙️ Status	In progress
Σ Week Day	Thursday → Monday
📌 Priority	High
➦ Project	📁 <u>TechTalk</u>
➦ Area	💖 <u>ITSector</u>

1. Nome da Apresentação;
2. Conteúdos (um sumário);
3. Alexandre Gonçalves **orador**, Diogo Moutinho, Emanuel Azevedo, Humberto Vieira.
4. Formato Híbrido

▼ Apresentação

- Apresentação dos Oradores
- Apresentação global da TechTalk;

▼ ReactJs

- O React é uma biblioteca de Javascript que facilita a criação de UIs interativas. Desenvolvido pelo Facebook com lançamento em 2013, tem vindo a ser um dos destaques mais influentes do mercado atual. [React History](#).
- A simplicidade do ReactJs é que criar um componente é tão simples como escrever uma função de Javascript. Ajuda no design de cada state da nossa aplicação. O React também vai eficientemente atualizar e renderizar somente os componentes desejados de acordo com certas alterações.

- O React é escrito com JSX que é uma extensão de sintaxe do JavaScript. É a junção de Javascript e XML, é basicamente a renderização do JavaScript mas com a estrutura do XML. O React está a ser compilado para dentro do React.CreateElement, o que basicamente mostra que o react está a criar elementos por debaixo dos panos. [Source](#)
- O data flow é uni-direcional, como o React tem uma relação de componentes pai para filho, passamos os dados de cima para baixo, isto através de props. [Source](#)
- O React trabalha com states internos, neste caso o useState, este retorna um valor e uma função para atualizar esse valor (counter → Read) e (setCounter → Write). Ao ser utilizado no template o counter vai sempre mostrar o seu valor mais recente. E sempre que o botão for clicado o react vai reagir (react) a esse evento.
- Neste contexto o Event Loop é responsável por gerenciar as atualizações da UI de forma eficiente. Quando uma ação é disparada, como o clique do botão, o React adiciona a tarefa de atualização à fila de eventos e aguarda a sua vez na fila para ser executada. Isso permite que outras tarefas importantes, como a renderização da interface, sejam realizadas sem serem interrompidas. Por fim o valor do counter vai ser reescrito, logo vai ser disparado um re-render desse estado e mostrado o valor mais atual do mesmo na DOM.

```
function App() {  
  const [counter, setCounter] = useState(0);  
  return (  
    <>  
      <p>{counter}</p>  
      <button onClick={() => setCounter(counter + 1)}>Click Me</button>  
    </>  
  );  
}
```

- A maior vantagem de se utilizar o React, não é tanto por conta do React em si, mas sim do enorme ecossistema que gira em torno dele. O React não se importa com

state management, routing, animações etc. mas esses conceitos evoluem naturalmente por conta da open source community.

- Sempre que acionado, o React cria uma reconstrução virtual da sua aplicação e compara-a com o estado real da sua aplicação (o DOM real) e atualiza apenas as partes que são diferentes, acelerando as atualizações, pois não está a reconstruir as partes que não mudaram. Portanto, quando essas mudanças são acionadas, é conhecido como uma renderização.
- Outra grande vantagem da utilização do React é a facilidade de transição para o ReactNative que nos permite começar a construir aplicações mobile.
- O Garbage Collector é uma característica do ReactJS que se encarrega de libertar memória de componentes que já não são necessários. Ele identifica componentes que foram descartados e remove as referências da memória, o que evita o acúmulo de "lixo" e otimiza o desempenho da aplicação.
- A curva de aprendizagem do React em comparação a outras frameworks é muito menor, isto deve-se ao facto da sintaxe do React ser clara e fácil de interpretar, para além da questão da performance ser muito mais eficiente que qualquer um dos concorrentes visto que o React faz uso da Virtual DOM que sempre que acionado, o React cria uma reconstrução virtual da sua aplicação e compara-a com o estado real da sua aplicação (o DOM real) e atualiza apenas as partes que são diferentes, acelerando as atualizações, pois não está a reconstruir as partes que não mudaram. Portanto, quando essas mudanças são acionadas, é conhecido como uma renderização.

▼ Inicialização de um projeto T3

Para a inicialização de um projeto utilizando a stack T3 é tão simples como correr o seguinte comando em um terminal:

```
npm create t3-app@latest
```

```
C:\Users\Alex\Desktop>npm create t3-app@latest

CREATE T3 APP

? What will your project be called? itsector-t3-tech-talk
? Will you be using TypeScript or JavaScript? TypeScript
Good choice! Using TypeScript!
? Which packages would you like to enable? nextAuth, prisma, tailwind, trpc
? Initialize a new git repository? Yes
Nice one! Initializing repository!
? Would you like us to run 'npm install'? Yes
Alright. We'll install the dependencies for you!
? What import alias would you like configured? @
```

Vão ser lançadas algumas questões para melhorar a customização como:

- Nome do Projeto;
- Se o Projeto vai fazer uso do Typescript ou uso do Javascript.
- Se o Projeto vai fazer uso de algum dos quatro packages integrados (nextAuth, Prisma, TailwindCss e tRPC);
- Se vai ser inicializado um novo repositório git;
- E se o projeto vai utilizar alguma alias nas importações do projeto.

▼ T3 Stack

Esta stack foi desenvolvida pelo [Theo](#) um Ex-desenvolvedor na Twitch e atual CEO da [Ping.gg](#) e criador de conteúdo, bastante conhecido por diversos comentários na plataforma X e Youtube.

A “T3 Stack” é uma stack de desenvolvimento web, que nada mais é que a combinação de linguagens de programação, frameworks, bibliotecas e ferramentas para facilitar o desenvolvimento e deploy de software. O objetivo da T3 é ser modular e typesafe sem comprometer a aplicação todas as ferramentas da stack são pensadas de modo a resolverem algum tipo de problema.

Esta tem como peças core o [Next.js](#), [Typescript](#) e [Tailwind CSS](#), estas 3 ferramentas são quase sempre incluídas em todos os projetos, já em certos casos em que esteja a ser desenvolvido algum tipo de backend a stack ainda conta com [tRPC](#), [Prisma](#) e [NextAuth.js](#).

▼ Next.js

O NextJs é utilizado por algumas das maiores empresas do mundo, esta ferramenta permite a criação de aplicações full-stack. Este é uma framework do ReactJs que utiliza “blocos de construção” para agilizar a criação de aplicações web.

Esta framework pode ser utilizada para desenvolvimento individual ou ser escalado numa grande equipa, a utilização dinamica do ReactJs e do Next.Js vai dar asas a criação de aplicações totalmente interativos, dinamicos e performaticos.

Blocos de Construção para uma Aplicação Web

- **User Interface** → Como é que utilizadores vão consumir e utilizar uma aplicação.
- **Routing** → Navegação entre diferentes partes da aplicação.
- **Data Fetching** → Onde a nossa data vive, onde a ir buscar e como a gerir.
- **Rendering** → Onde e como renderizar conteudo estatico ou dinamico.
- **Integrations** → Integração com outras ferramentas, (Auth, payments, prisma).
- **Infrastructure** → Onde dar deploy, armazenar e correr a aplicação (Serverless, CDN, Edge, etc).
- **Performance** → Optimização da aplicação para os utilizadores.
- **Scalability** → Adaptação e evolução da aplicação a medida em que a equipa/projeto crescem.


Pode ser utilizado o ReactJS para construir o UI da aplicação e ir adotando as ferramentas que o Next.JS proporciona para resolver os problemas comuns do ReactJS, como Routing, Data Fetching, Integrations.

```
// mostrar routing live, criar uma pagina e navegar para a mesma.
```

Next.js in 100 Seconds // Plus Full Beginner's Tutorial

Learn the basics of Next.js in 100 Seconds! Then build your first server-rendered react app with a full Next.js beginner's tutorial.

<https://fireship.io/courses/react-next-firebase>

 https://www.youtube.com/watch?v=SkIc_fQBmcs&ab_channel=FireShip



<https://nextjs.org>

<https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>


▼ Typescript


<https://www.typescriptlang.org>

O Typescript é um superset, um conjunto de ferramentas, montado em cima do Javascript que adiciona tipagem estática, parâmetros, retornos em cima do Javascript.


Como muitos sabem, o Javascript é uma terra sem grandes regras, podemos referenciar variáveis que não existem ou até mesmo trabalhar com objetos sem forma definida, o código é então interpretado por um browser mas caso existam algum erro só vai ser descoberto quando a aplicação estiver a correr, o Typescript previne destes erros de sequer acontecerem estendendo o Javascript com tipagens.


```
//Javascript 🗨️  
randomObj.random()
```

IDE -> 

RUN ->  Uncaught ReferenceError: randomObj is not defined at X:2:1

```
//Typescript 👍  
randomObj.random()
```

IDE ->  Cannot find name "randomObj". ts(2304)

RUN ->  Cannot find name "randomObj". ts(2304)

O Typescript permite também, através do ficheiro tsconfig, criar uma experiência única de utilização para cada projeto e desenvolvedor modificando as regras de utilização do mesmo. Mas a sua funcionalidade principal é a tipagem estática que conta com explicit e implicit types

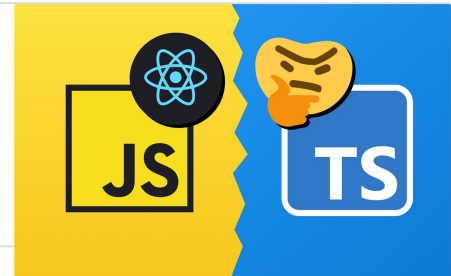
```
let name: string; //Explicit Type
```

```
let surname = "Gonçalves" //Implicit Type
```

How to use TypeScript with React... But should you?

Learn how to setup React with TypeScript. Compare the pros and cons of using TypeScript in an React project. Learn more in the full course 🖱️ <https://fireship.io/courses/react-next-firebase/>

📺 https://www.youtube.com/watch?v=ydkQIJhodio&ab_channel= Fireship



Criar exemplo no código na área de Typescript onde demonstro que o meu component agora pode receber todas as props do button. Mostrar também que o name é required e que o Typescript vai gritar comigo caso não o use

```
import {ComponentProps} from "react"

type ButtonProps = {
  name:string // example
} & ComponentProps<'button'>

/* OR

interface ButtonProps {
  name:string // example
} extends React.ButtonHTMLAttributes<HTMLButtonElement>
*/

export const ButtonWithProps = ({name, children, ...props}:ButtonProps) => {
  return <button data-name={name} {...props}>{children ?? "Button with props"}</button>
}

export const ButtonWithoutProps = () => {
  return <button>Button without props</button>
}
```

Generic Type example

// <https://www.rst.software/blog/advanced-typing-in-typescript-with-generics>

▼ Tailwind CSS

O Tailwind CSS é uma framework de design para criação de UI responsivas e modernas em projetos web. O Tailwind, ao contrário do Bootstrap que oferece componentes, fornece classes utilitárias de baixo nível que podem ser aplicadas diretamente aos elementos HTML para estilizar o conteúdo de maneira eficiente e

flexível que cobrem desde espaçamento e tipografia até cores e posicionamento. Isso permite aos desenvolvedores personalizarem o estilo dos seus componentes muito mais eficientemente.

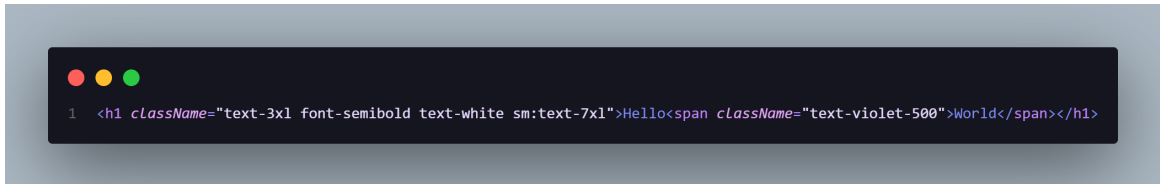
```
1 import { type Config } from "tailwindcss";
2
3 export default {
4   content: ["/src/**/*.js,ts,jsx,tsx"],
5   theme: {
6     extend: {
7       fontFamily: {
8         Figtree: "Figtree, sans-serif",
9       },
10      boxShadow: {
11        inset: "inset 0px 0px 41px rgba(0,0,0,0.55)",
12        navbar: "0px 1px 3px 0px rgba(0,0,0,0.10)",
13      },
14    },
15  },
16  plugins: [
17    // eslint-disable-next-line @typescript-eslint/no-unsafe-call, @typescript-eslint/no-var-requires
18    require("tailwindcss-accent")({
19      colors: ["violet", "blue", "orange"],
20      root: "violet",
21    }),
22    // ...
23  ],
24 } satisfies Config;
```

As principais características do Tailwind CSS incluem:

1. **Classe Utilitárias** → A framework oferece uma variedade de classes utilitárias que podem ser aplicadas diretamente aos elementos.
2. **Configurável** → O Tailwind é altamente configurável. Os developers podem personalizar as classes já existentes no tailwind ou estender os seus próprios estilos para facilitar no desenvolvimento da aplicação.
3. **Responsividade** → A framework possui classes responsivas que permitem ajustar a estilização dependentemente do tamanho do ecrã no qual a aplicação esta a ser renderizada.
4. **Plugins** → Os plugins do Tailwind permitem estender as funcionalidades da framework. Existem diversos plugins criados pela comunidade que adicionam recursos extras, como estilos de terceiros, animações e componentes.

5. **Bibliotecas de componentes** → Embora o Tailwind CSS não ofereça componentes prontos a utilizar, é possível usar algumas bibliotecas de componentes tailwind como o [TailwindUI](#) ou a [DaisyUI](#).

Em resumo, o Tailwind CSS é uma ferramenta poderosa para desenvolvedores que buscam criar designs personalizados de forma eficiente, sem escrever um grande volume de CSS.



▼ Prisma

Prisma é uma ORM que corre no server-side e aumenta a produtividade e experiencia do desenvolvedor ao gerar funções (depois do setup, npx prisma db push) que seriam usadas para ir buscar dados à base de dados, sem ter que escrever SQL ou queries em MongoDB, por exemplo.

```
// schema.prisma (config main file)

generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider      = "mysql"
  url           = env("DATABASE_URL")
}

model User {
  id          String    @id @default(uuid())
  name        String?
  image       String?
  email       String?   @unique
  emailVerified DateTime?
```

```

posts      Post[]
}

model Post {
  id        String    @id @default(uuid())
  userId    String
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  user User @relation(fields: [userId], references: [id])

  @@index([userId])
}

```

▼ NextAuth.JS

O NextAuth.Js é uma excelente solução que adiciona um complexo nível de segurança sem o esforço de terem de ser desenvolvida de raiz. Vem com uma extensa lista de providers para rapidamente construir um autenticação OAuth assim como uma grande lista de adaptars para se integrar com diversas bases de dados e ORMs.

As principais características do NextAuth.JS incluem:

- Desenhado para funcionar com qualquer serviço OAuth e OpenID.
- Suporte integrado com varios serviços de sign-in.
- Suporta autenticações sem email/password.
- Suporta tanto com **JSON web token** como sessões de **bases de dados**.
- Suporte integrado para **MySQL, MariaDB, Postgres, SQL Server, MongoDB and SQLite**.
- Desenhado para ser seguro, encoraja as melhores práticas para salvaguardar dados dos utilizadores.

▼ tRPC