

A screenshot of a terminal window. On the left, there's a file tree for a project named 'Coding'. It includes folders like '.git', 'FAQ.md', 'log.md', 'ri-log.md', 'README.md', 'resources.md', 'rules.md', 'atom-packages', 'browser_persistence', 'c01', 'FlashcardsExpress', 'frescodescamp_tribute', 'JavaScript-Authentication', 'LocalWeatherFCC', 'node-weather-zipcode', 'nodeschool', 'nodeWeather', 'portfolio', and 'the-vow-test'. There are also files like 'index.js', 'app.js', and 'package.json'. On the right side of the terminal, there are two code snippets. The top one is from 'index.js' and shows a function for handling a 'register' request. The bottom one is from 'routes/index.js' and shows a function for creating a user.

```
Project
  Coding
    100-days-of-code
      .git
      FAQ.md
      log.md
      ri-log.md
      README.md
      resources.md
      rules.md
      atom-packages
      browser_persistence
      c01
      FlashcardsExpress
      frescodescamp_tribute
      JavaScript-Authentication
        .git
        models
        public
      routes
        index.js
      views
      .ignore
      app.js
      package.json
      README.md
    LocalWeatherFCC
    node-weather-zipcode
    nodeschool
    nodeWeather
    portfolio
    the-vow-test
  index.js
  app.js
  package.json
  README.md

  // GET /register
  router.get('/register', function(req, res, next) {
    return res.render('register', { title: 'Sign Up' });
  });

  // POST /register
  router.post('/register', function(req, res, next) {
    var user = new User({
      email: req.body.email,
      name: req.body.name,
      favoriteBook: req.body.favoriteBook,
      password: req.body.password
    });

    user.save(function(err) {
      if (err) {
        return next(err);
      }
      // Create object with form input
      var(userData = {
        email: req.body.email,
        name: req.body.name,
        favoriteBook: req.body.favoriteBook,
        password: req.body.password
      });

      // use schema's create
      User.create(userData, function(error) {
        if (error) {
          return next(error);
        }
      });
    });
  });

  module.exports = router;
```

COM AMOR

STYLED COMPONENTS



TYPESCRIPT

REACT NATIVE

IT'S ME, HROAD

MATEUS + PEDRO <3



HELENA STRADA

Engenheira de Software

```
  al';
  extractJwt } from 'passport-jwt';

  );
  const { User, auth, helper };

  'name' );

  where('username', username);

  user(userData);
  if (password) {
    }

  /**
   * JWT Strategy Auth
   */
  const jwtOpts = {
    // Telling Passport to check authorization headers for JWT
    jwtFromRequest: ExtractJwt.fromAuthHeaderWithScheme('JWT'),
    // Telling Passport where to find the secret
    secretOrKey: constants.JWT_SECRET,
  };

  const jwtLogin = new JWTStrategy(jwtOpts, async (payload, done) => {
    try {
      console.log(payload);
      const user = await User.query().where('user_uuid', payload.user_uuid);
      console.log(user[0].toJSON());

      if (user.length === 0 || !user) {
        return done(null, false);
      }

      return done(null, user[0]);
    }
  });
}

export default jwtLogin;
```



EU PENSEI NOS SEGUINTE TÓPICOS

Introdução

- Atual cenário
- Problemas Encontrados (com exemplos de código)

Proposta

- Styled Components
- TypeScript
- Nosso Desafio ~ não falar sobre o que é o app
- Desenvolvendo a Solução Proposta (com exemplos de código)

Resumo

- Prós - qtd de requisições na web
- Cons
- Comentar sobre a nossa experiência
- Tempo de desenvolvimento um pouco maior no início para a criação dos componentes

Contexto

- O que escolher?

O que soluciona melhor seu problema ~ cuidado com over engineering

Custo (em tempo ou \$ ~ caso houvesse)

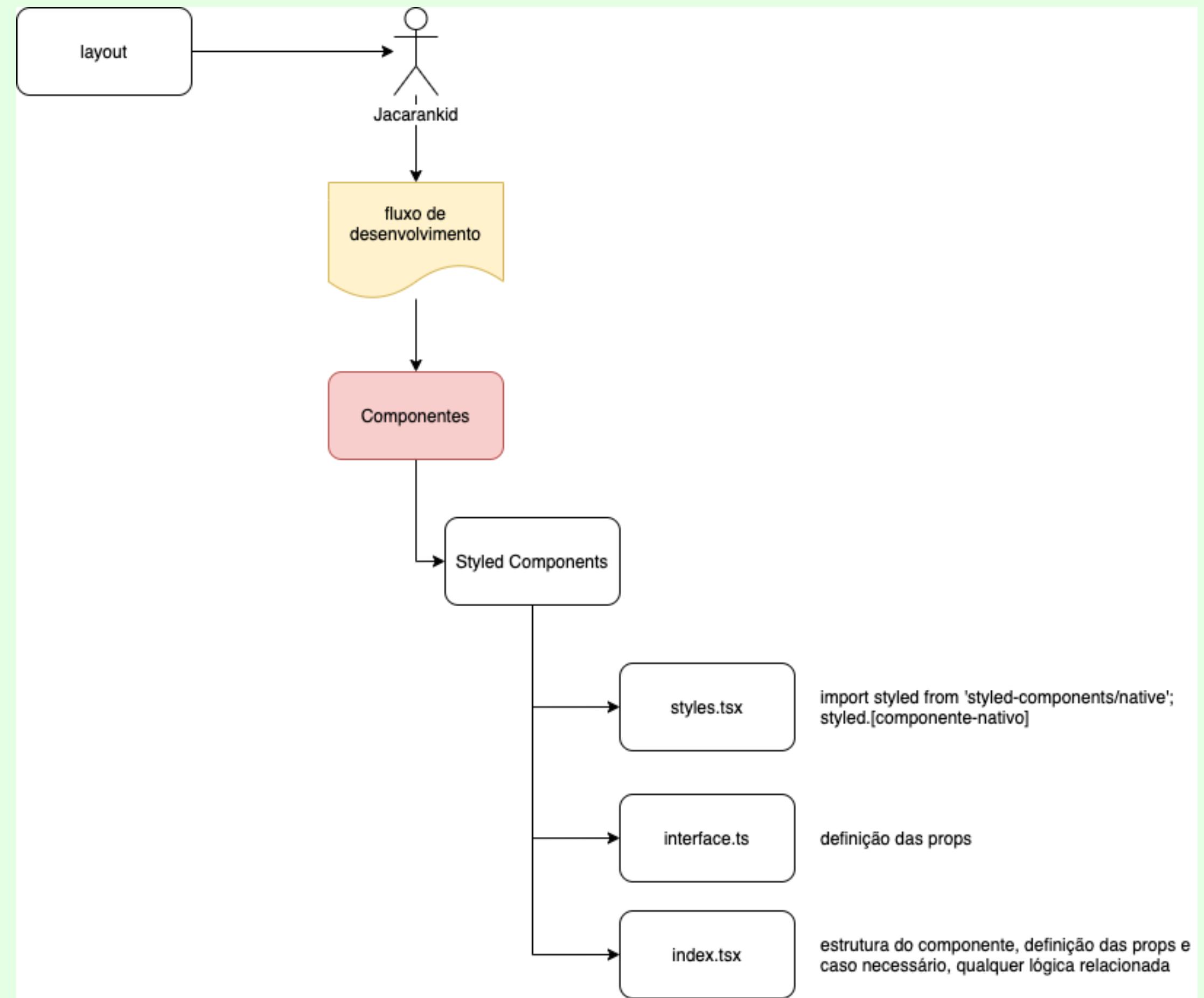
O que a sua equipe conhece ~ experiência

O que a sua equipe está disposta a conhecer ~ engajamento/desafios

Tempo até a entrega do projeto ~ curva de aprendizagem

Sobre tecnologia: não se apegar a ferramentas

ter ferramentas que auxiliem na solução de problemas



CONTEXTO

REACT NATIVE

Mobile elegante.

TYPESCRIPT

Tipagem, interfaces, recursos e muito amor.

CSS-IN-JS (STYLED COMPONENTS)

Sem classname, escopo do componente.

STYLED COMPONENTS

**styled-components is the result
of wondering how we could
enhance CSS for styling React
component systems.**

[HTTPS://STYLED-COMPONENTS.COM/DOCS/BASICS](https://styled-components.com/docs/basics)



SHOW ME THE CODE O INÍCIO

```
● ● ●  
  
import styled from 'styled-components/native';  
  
export const Container = styled.TouchableOpacity``;  
  
export const Label = styled.Text``;
```

styles.tsx

```
● ● ●  
  
import React from 'react';  
  
import { Container, Label } from './styles';  
  
const FlatButton = () => {  
  return (  
    <Container>  
      <Label>Hello World</Label>  
    </Container>  
  );  
};  
  
export default FlatButton;
```

index.tsx

src/components

SHOW ME THE RESULT

Hello World



```
import React from 'react';
import { SafeAreaView } from 'react-native';

import { FlatButton } from './src/components/';

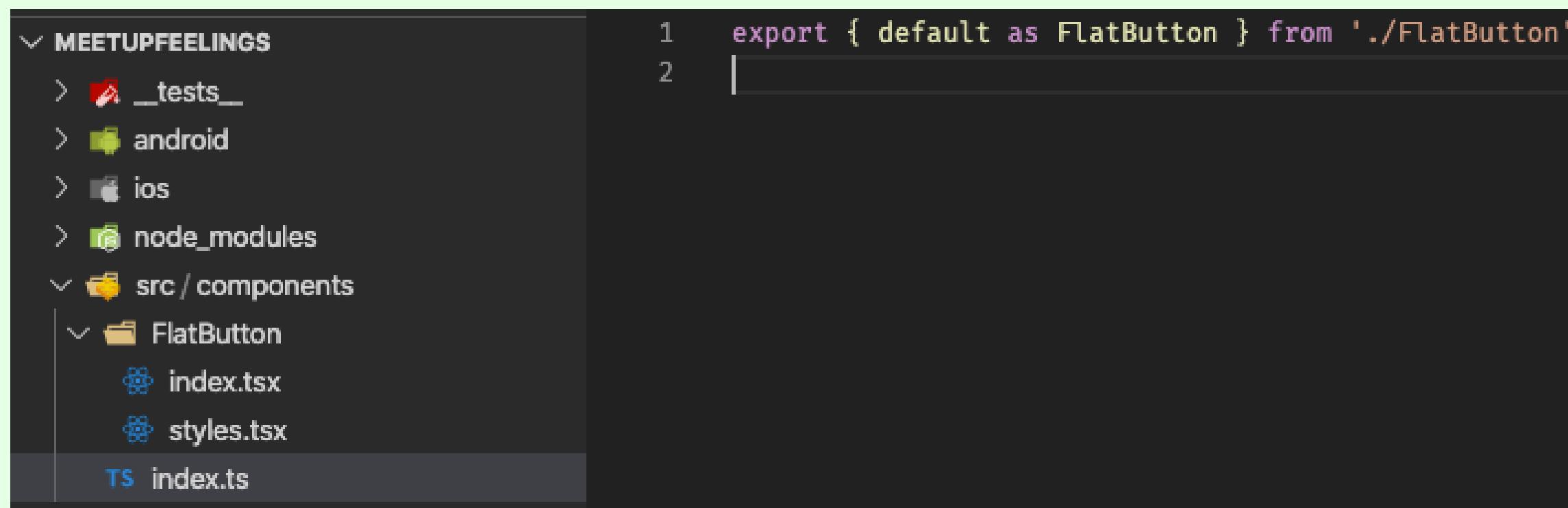
const App = () => {
  return (
    <SafeAreaView>
      <FlatButton />
    </SafeAreaView>
  );
};

export default App;
```

App.tsx

Cuidado somente ao criar os nomes de componentes.

Normalmente deixamos eles distintos dos components nativos para que não aja nenhuma troca ou como se fosse uma reescrita do mesmo.



The screenshot shows a file tree on the left and a code editor on the right. The file tree includes: MEETUPFEELINGS, __tests__, android, ios, node_modules, src / components (which contains FlatButton, index.tsx, and styles.tsx), and index.ts. The code editor has two lines of code:

```
1 export { default as FlatButton } from './FlatButton';
2 |
```

src/components/index.ts

COMO ESTRUTURAMOS?

AUMENTANDO A COMPLEXIDADE E SEPARANDO RESPONSABILIDADES

index.tsx

Contém o desenho de nossa estrutura do componente.

styles.tsx

Contém nossos componentes.

interface.ts

Possui tanto as props que o componente irá receber, como as props que serão enviadas ao estilo.

- PROPS, SUA LINDA.

```
● ● ●  
export interface Props {  
  text: string;  
}
```

src/components/FlatButton/interface.ts



```
import React from 'react';  
  
import { Props } from './interface';  
  
import { Container, Label } from './styles';  
  
const FlatButton = ({ text }: Props) => {  
  return (  
    <Container>  
      <Label>{text}</Label>  
    </Container>  
  );  
};  
  
export default FlatButton;
```

.../FlatButton/index.ts

RECLAMONA

```
const App = () => {
  return (
    <SafeAreaView>
      <FlatButton />
    );
};

export [REDACTED]

const FlatButton = ({ text }: Props) => {
  return (
    <Container>
      <Label>{text}</Label>
    </Container>
  );
};

(alias) const FlatButton: ({ text }: Props) => JSX.Element
import FlatButton

Property 'text' is missing in type '{}' but required in type 'Props'. ts(2741)
interface.ts(2, 3): 'text' is declared here.

Peek Problem (F8) No quick fixes available
```

NÃO MAIS RECLAMONA



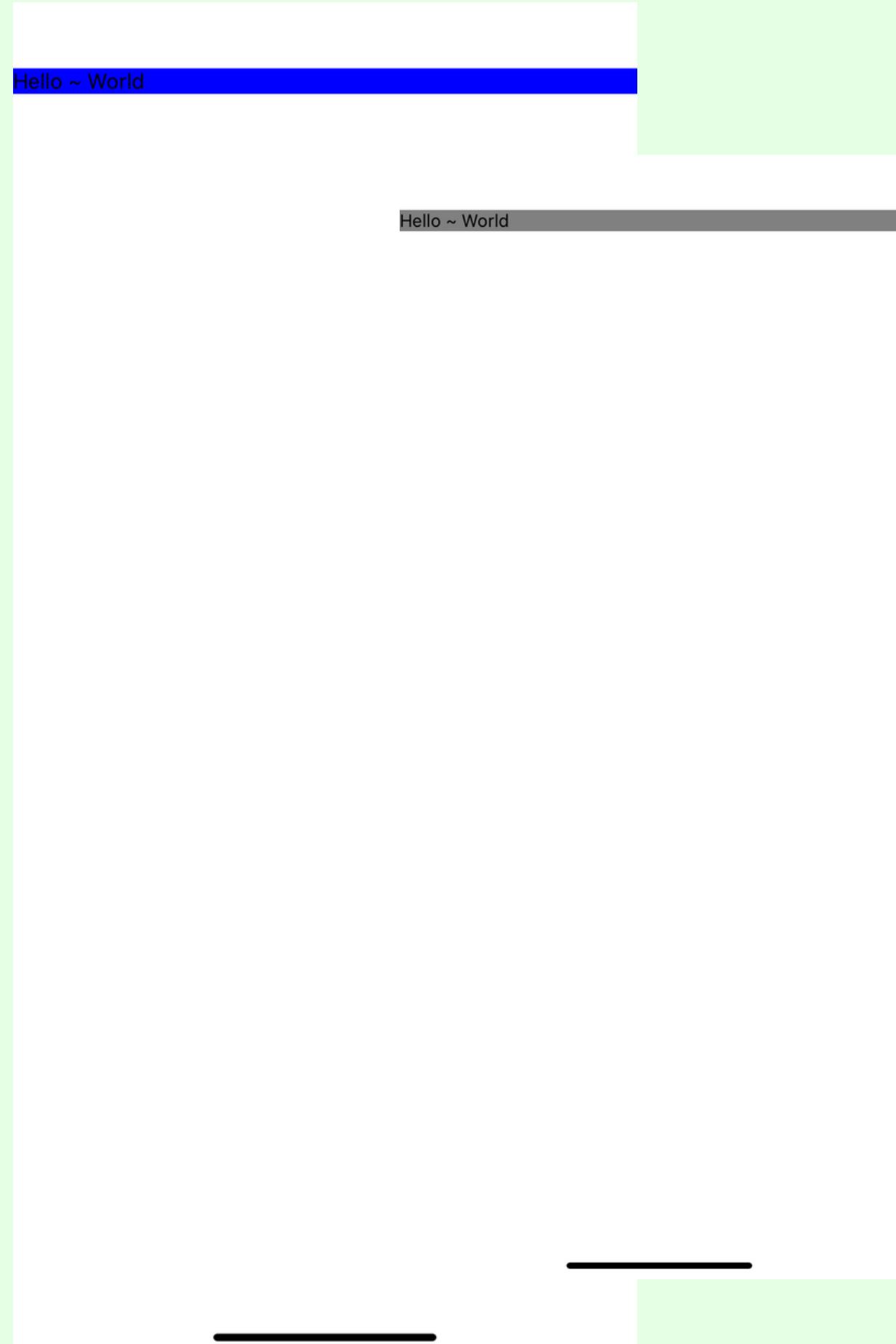
CONDICIONAIS

```
● ● ●  
import React from 'react';  
  
import { Props } from './interface';  
  
import { Container, Label } from './styles';  
  
const FlatButton = ({ text, disabled }: Props) => {  
  return (  
    <Container disabled={disabled}>  
      <Label>{text}</Label>  
    </Container>  
  );  
};  
  
export default FlatButton;
```

index.tsx

interface.ts

```
● ● ●  
export interface Props {  
  text: string;  
  disabled?: boolean;  
}  
  
export interface PropsStyled {  
  disabled?: boolean;  
}
```



styles.tsx

```
import styled from 'styled-components/native';

import { PropsStyled } from './interface';

export const Container = styled.TouchableOpacity<PropsStyled>`  
  background-color: ${({props}) => (props.disabled ? 'gray' : 'blue')};  
`;  
  
export const Label = styled.Text``;
```

A screenshot of a code editor window. The title bar shows three colored dots (red, yellow, green). The main area contains the following TypeScript code for a styled component:

MELHORANDO A LEGIBILIDADE

DANDO UM AMOR

App.tsx

```
import { FlatButton } from './src/components/';
const App = () => {
  return (
    <SafeAreaView>
      <FlatButton text="Hello ~ World" disabled={true} />
    </SafeAreaView>
  );
}

export default App;
```

```
import React from 'react';
import { Props } from './interface';
import { Container, Label } from './styles';

/**
 ...
 Example: <FlatButton text="Hello World" disabled={true} />
 ...
 */
const FlatButton = ({ text, disabled }: Props) => {
  return (
    <Container disabled={disabled}>
      <Label>{text}</Label>
    </Container>
  );
};

export default FlatButton;
```

FlatButton/index.tsx

AQUI EU IREI INCLUIR

ITENS EXTRAS, COMO TEMAS,
VARIANTS, ETC.

BENEFÍCIOS PROS AND CONS

- A curva de desenvolvimento no início é um pouco maior (criação de componentes); Depois acaba se tornando menor.
- Evitar requisições a arquivos externos (web);
- Escalabilidade;
- Manutenção sem 'side-effects' ~ um único ponto de manutenção;
- Estilos dinâmicos são mais claros de controlar;
- Definição de um tema ajuda e muito em caso de alterações;

RESUMINDO: O QUE ESCOLHER?

- O que a sua equipe tem experiência?
- O que a sua equipe está disposta a conhecer/desenvolver?
- Custo (em tempo ou \$ ~ caso houvesse)
- Tempo até a entrega do projeto ~ curva de aprendizagem
- Escalabilidade do projeto

IDEIAS

O que resolve seu problema?

O que melhor te auxilia.

Não se apegar a algo específico.

Curtir e dar amor pro seu código.

The screenshot shows a code editor with a file named 'Servers.js' open. The file is part of a React Native application structure. The code imports various components from 'react-native' and other local modules like 'lodash' and 'firebase'. A large portion of the code is obscured by a red rectangular redaction box. The file is saved at the path 'app/screens/Servers.js'.

```
Project
minecraftPeServersApp
  __tests__
  .git
  android
  app
    assets
    components
    config
    screens
      About.js
      Favorites.js
      FilterServers.js
      News.js
      Server.js
      Servers.js
      Submit.js
      Tutorial.js

Servers.js — ~/Documents/Personal-Projects/minecraftPeServersApp
  JS Server.js
  JS globals.js
  JS Submit.js

1 import React, { Component } from 'react';
2 import {
3   TouchableHighlight,
4   RefreshControl,
5   Platform,
6   AppState
7 } from 'react-native';
8 import { Toast, Card, CardItem, Container,
9 import { Col, Grid } from 'react-native-ea
10 import Icon from 'react-native-vector-icon
11 var _ = require('lodash');
12 import firebase from 'react-native-firebase
13
14 import ServerComp from '../components/Se
15 import Data from '../utilities/Data';
16 import Cache from '../utilities/Cache';
17
18
19
20
21
22
23
24
```

o se apegar a algo
específico.

Curtir e dar amor pro
seu código.