React Native

Desenvolvimento de Software e Sistemas Móveis (DSSMV)

Licenciatura em Engenharia de Telecomunicações e Informática

LETI/ISEP

2025/26

Paulo Baltarejo Sousa

pbs@isep.ipp.pt

Instituto Superior de Engenharia do Porto

P.PORTO

Disclaimer

Material and Slides

Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

Outline

Setup

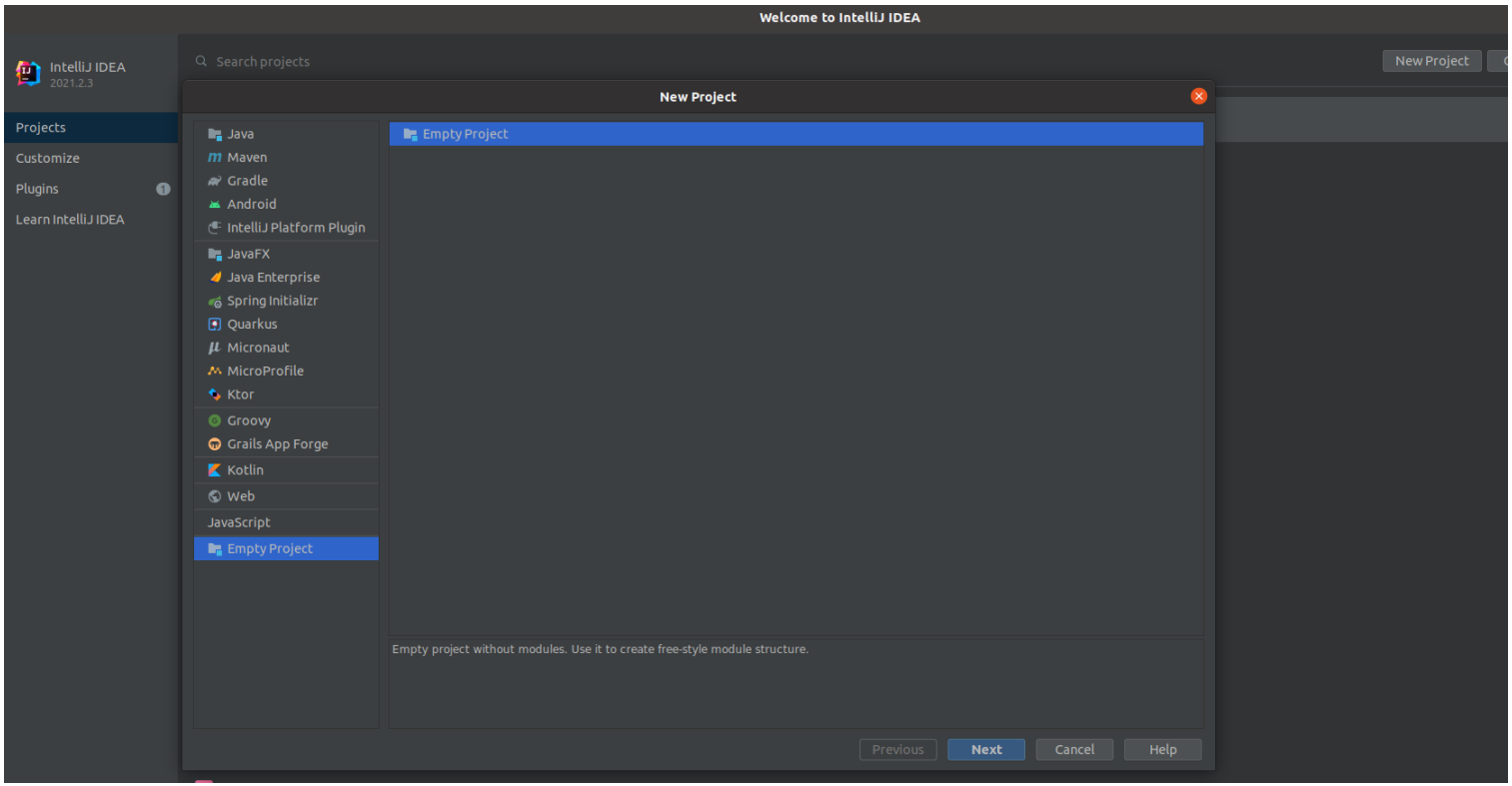
Links

- Setting up the development environment
  - `https://reactnative.dev/docs/environment-setup`
- Running On Device
  - `https://reactnative.dev/docs/running-on-device`

Let's start

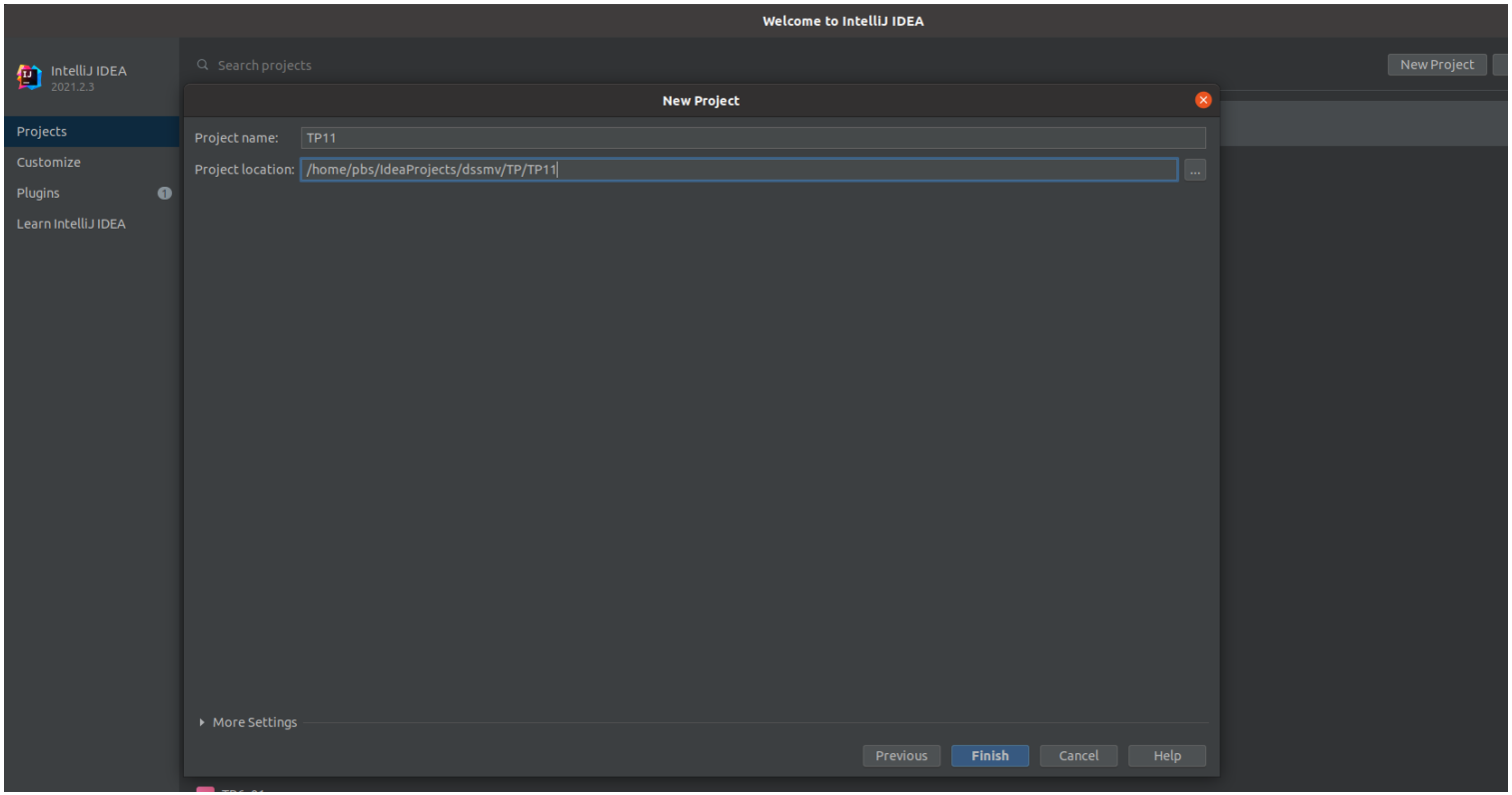
Create an Empty Project (I)

- Open IntelliJ IDEA, select `New Project > Empty Project.`
- Click on `Next`



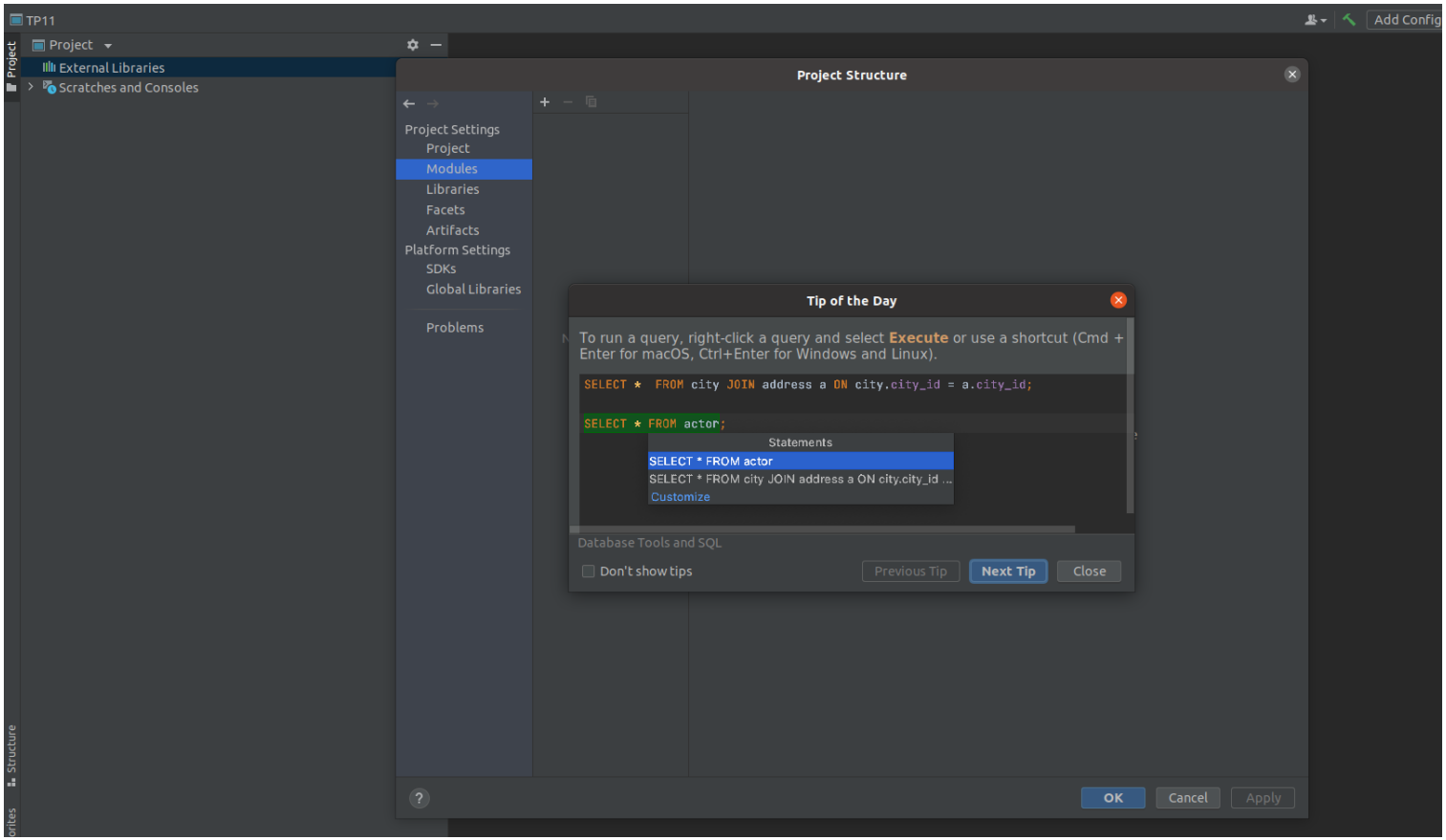
Create an Empty Project (II)

- Name it and click on `Finish`



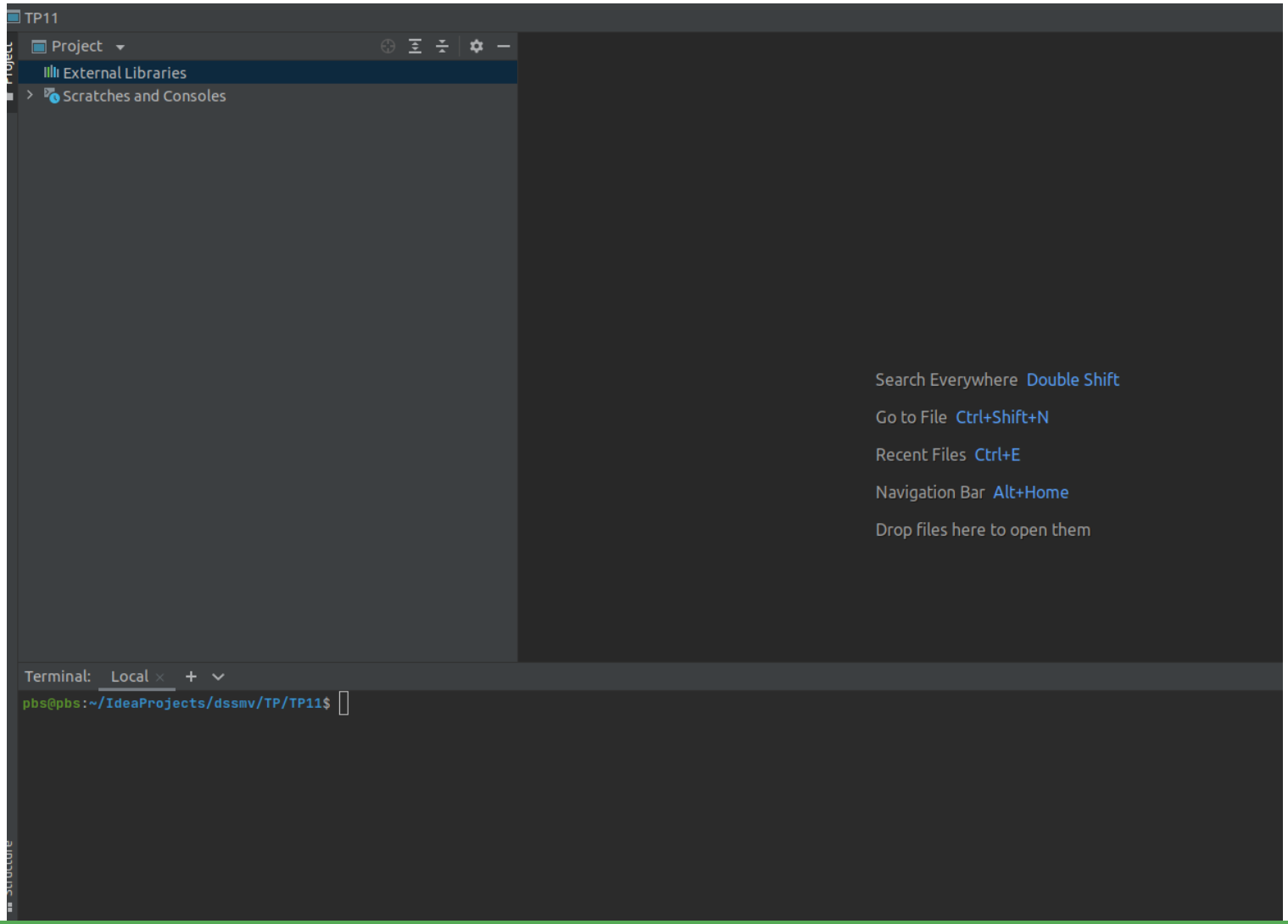
Create an Empty Project (III)

- Click the `Close` button in the `Tip of the Day` dialog.
- Click on `Cancel` in the `Project Structure` window.



Create an Empty Project (IV)

- In the Intellj IDEA, open the Terminal Window



Create React Native App

- Create an app

```
npx @react-native-community/cli@latest init
FirstApp
cd FirstApp
```

- Set the emulator (Fixing errors)

```
npx react-native doctor
```

- Launch Metro

```
npm start
```

- Launch app in Android device (in another terminal)

```
npm run android
```

Create React App: Structure (I)

- Folders:
  - `node_modules`: Javascript library
  - `android`: Android project.
  - `ios`: iOS project.
  - `__tests__`: Tests.
- JavaScript files:
  - `App.js`: The main component.
  - `index.js`: The main file of our application where the components are registered.
- Configuration files
  - `package.json` and `package-lock.json` : Node.js configuration files.
  - `.gitignore` and `.gitattributes`: for git.

Create React App: Structure (II)

- Configuration files (cont.)
  - `.prettierrc.js`: Code formatter.
  - `babel.config.js`: The configuration file for Babel (A compiler and transpiler for JavaScript)
  - `metro.config.js`: The configuration file for Metro, a JavaScript bundler for React Native,
  - `app.json`: it is used to configure many things such as app name, icon, splash screen, deep linking scheme, API keys to use for some services and so on.
  - `watchmanconfig`: The configuration file for Watchman, a file watch service. This service watches files and records when they change. It can also trigger actions (such as rebuilding assets) when matching files change.
  - `.eslintrc.js`: The configuration file for ESLint, a JavaScript and JSX linter (a tool for code quality).
  - `tsconfig.json` file specifies Typescript project configuration.
  - `Gemfile` is a file that is created to describe the `gem` dependencies required to run a Ruby program.

Entry point

- `index.js`

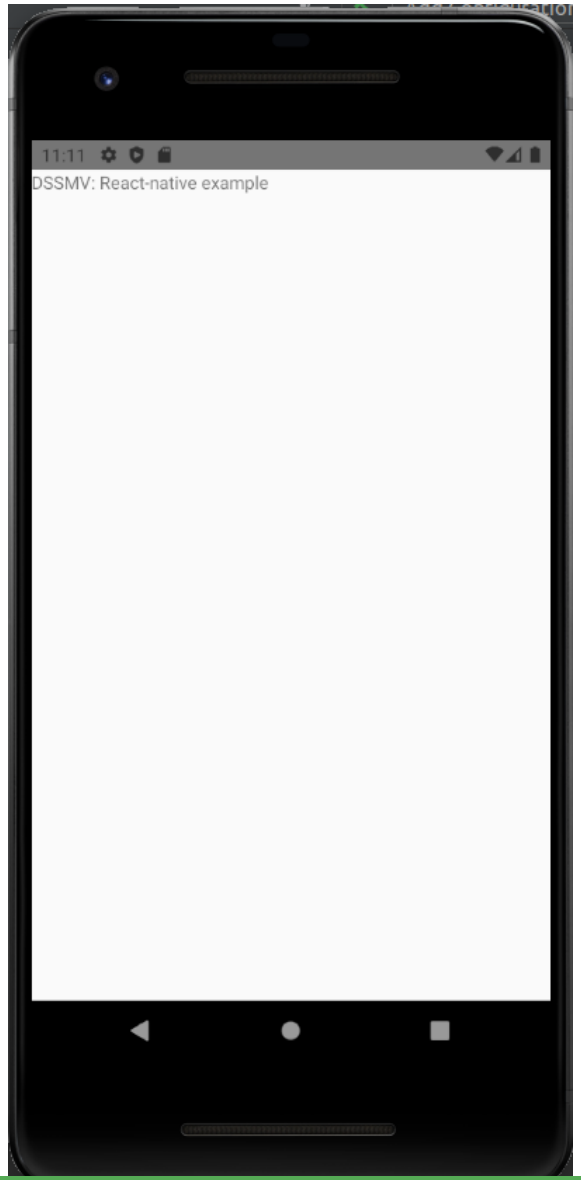
```
import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

  - `AppRegistry` is the JS entry point to running all React Native apps.
  - App root components should register themselves with `AppRegistry.registerComponent`
  - The native system can load the bundle for the app and then actually run the app when it's ready by invoking `AppRegistry.runApplication`.

FirstApp Example: First iteration

Output



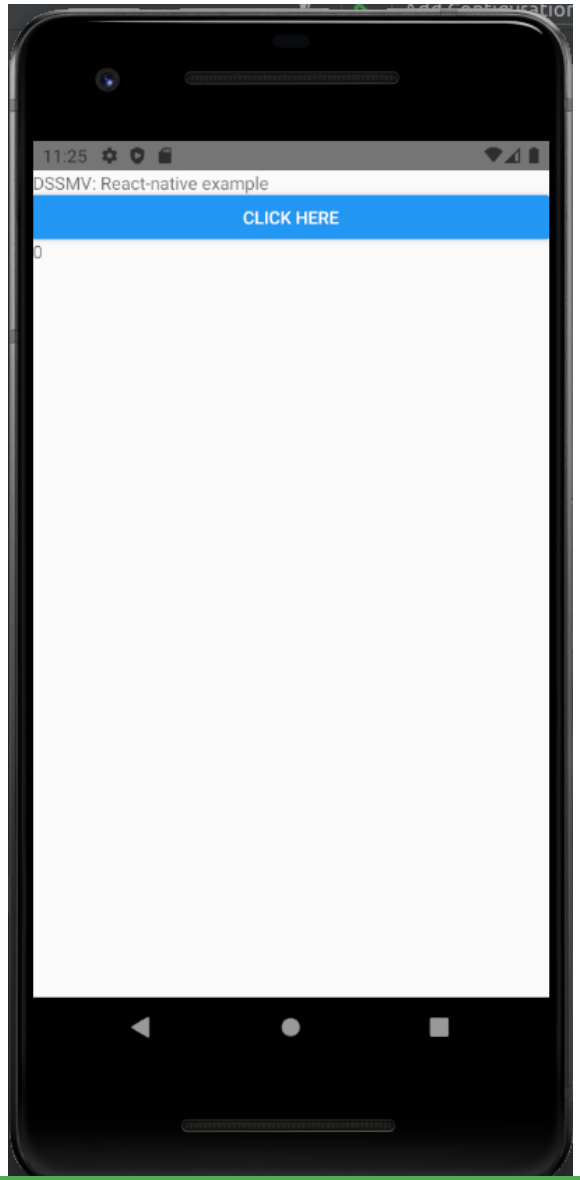
App component

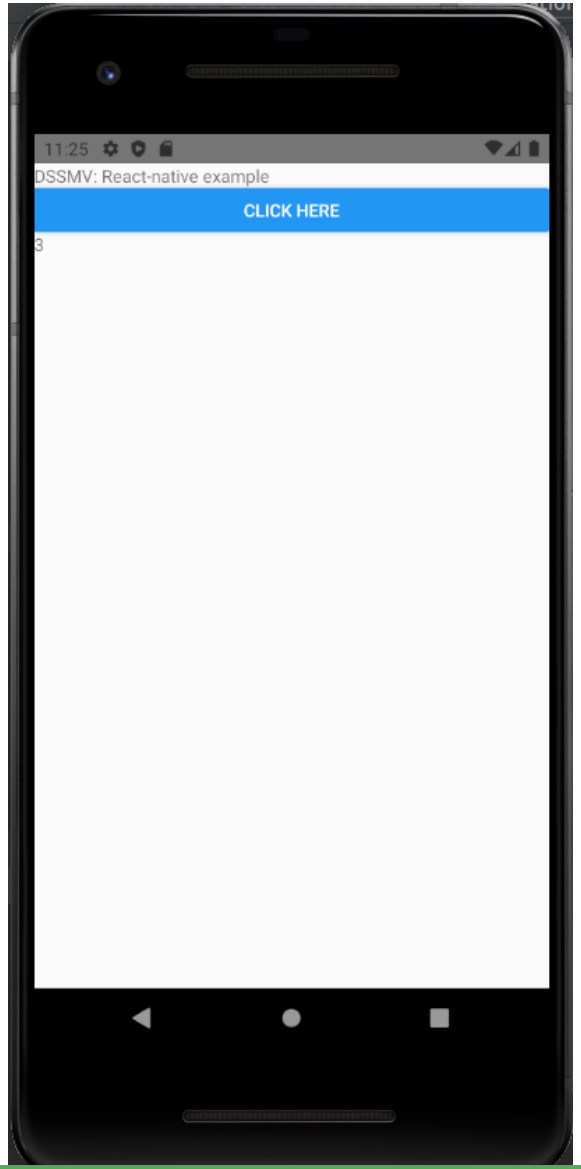
- Changes on App.js file

```
import React, {Component} from 'react';
import {View, Text} from 'react-native';
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      title: 'DSSMV: React-native example',
    };
  }
  render() {
    const title = this.state.title;
    return (
    <View>
      <Text>{title}</Text>
    </View>
    );
  }
}
export default App;
```

FirstApp Example: Second iteration

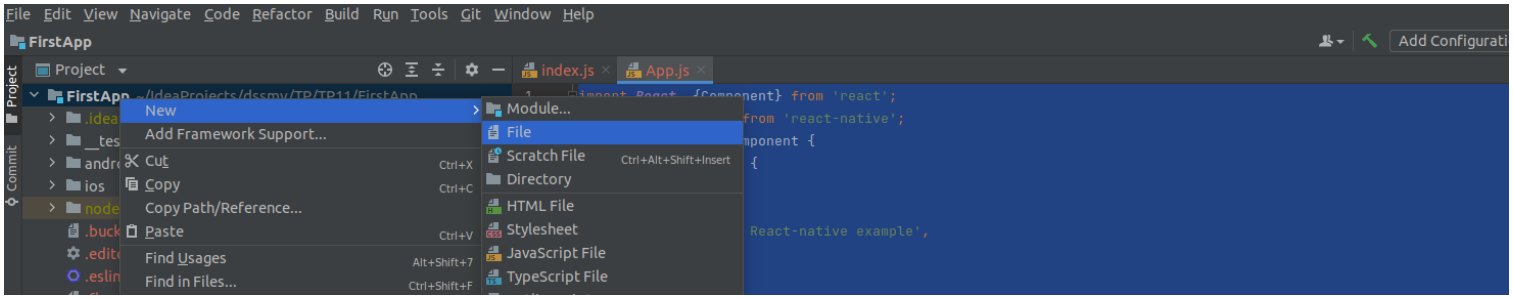
Output


11:25
DSSMV: React-native example
CLICK HERE
0

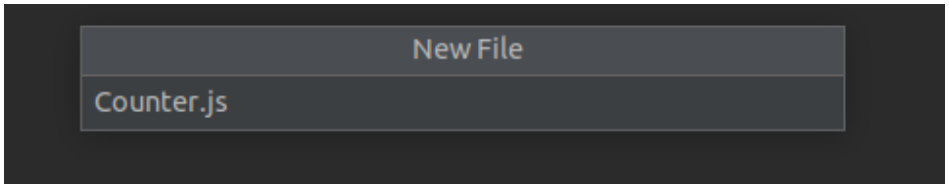

11:25
DSSMV: React-native example
CLICK HERE
3

`Counter` component (I)

- Right click on `FirstApp > New > File`



- Name it `Counter.js`



Counter component (II)

```
import React, {Component} from 'react';
import {View, Text, Button} from 'react-native';
class Counter extends Component {
  constructor(props) {
    super(props);
    this.state = {count: 0, };
  }
  handleClick = () => {
    let val = this.state.count;
    val = val + 1;
    this.setState({count: val});
  };
  render() {
    const count = this.state.count;
    return (
    <View>
      <Button onPress={this.handleClick} title="Click here" />
      <Text>{count}</Text>
    </View>
    );
  }
}
export default Counter;
```

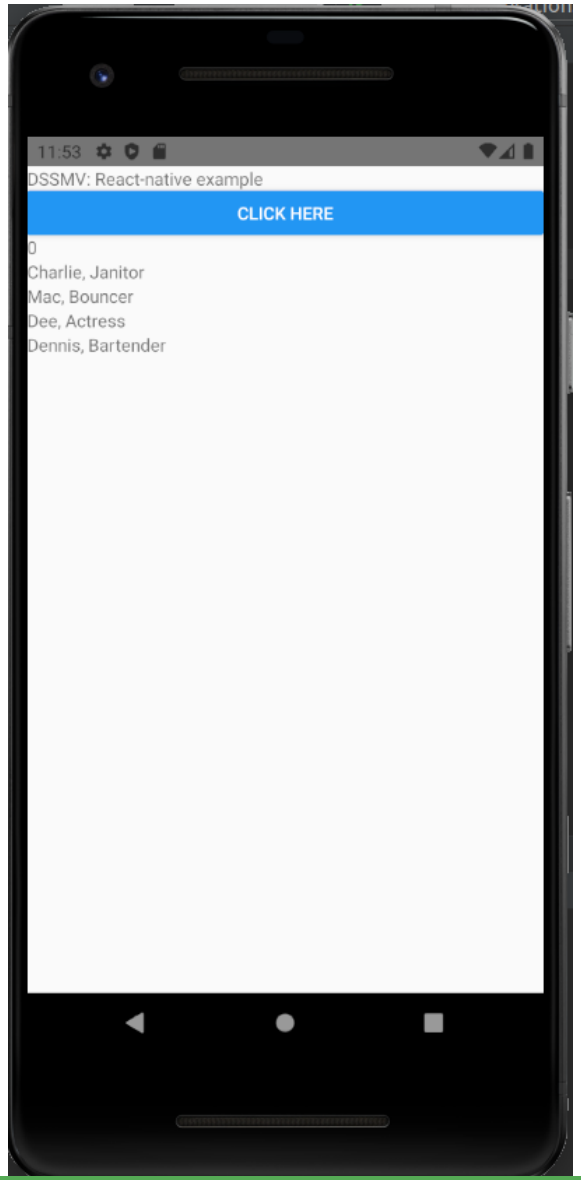
App component

- Changes on App.js file

```
import React, {Component} from 'react';
import {View, Text} from 'react-native';
import Counter from './Counter';
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      title: 'DSSMV: React-native example',
    };
  }
  render() {
    const title = this.state.title;
    return (
    <View>
      <Text>{title}</Text>
      <Counter />
    </View>
    );
  }
}
export default App;
```

FirstApp Example: Third iteration

Output


11:53
DSSMV: React-native example
CLICK HERE
0
Charlie, Janitor
Mac, Bouncer
Dee, Actress
Dennis, Bartender

PeopleListItem component

- Create PeopleListItem.js file

```
import React, {Component} from 'react';
import {View, Text} from 'react-native';

class PeopleListItem extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    const {name, job} = this.props;
    return (
    <View>
      <Text>
        {name}, {job}
      </Text>
    </View>
    );
  }
}
export default PeopleListItem;
```

PeopleList component

- Create `PeopleList.js` file

```
import React, {Component} from 'react';
import {FlatList, Text} from 'react-native';
import PeopleListItem from './PeopleListItem';

class PeopleList extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    const people = this.props.listOfItems;
    return (
      <FlatList
        data={people}
        keyExtractor={item => item.id}
        renderItem={({item}) => (
          <PeopleListItem key={item.id} name={item.name} job={item.job} />
          )}
      />
    );
  }
}
export default PeopleList;
```

App component (I)

- Changes on App.js file

```
import React, {Component} from 'react';
import {View, Text} from 'react-native';
import Counter from './Counter';
import PeopleList from './PeopleList';
const data = [
  {name: 'Charlie', job: 'Janitor'},
  {name: 'Mac', job: 'Bouncer'},
  {name: 'Dee', job: 'Aspring actress'},
  {name: 'Dennis', job: 'Bartender'},
];
class App extends Component {
  ...
}
export default App;
```

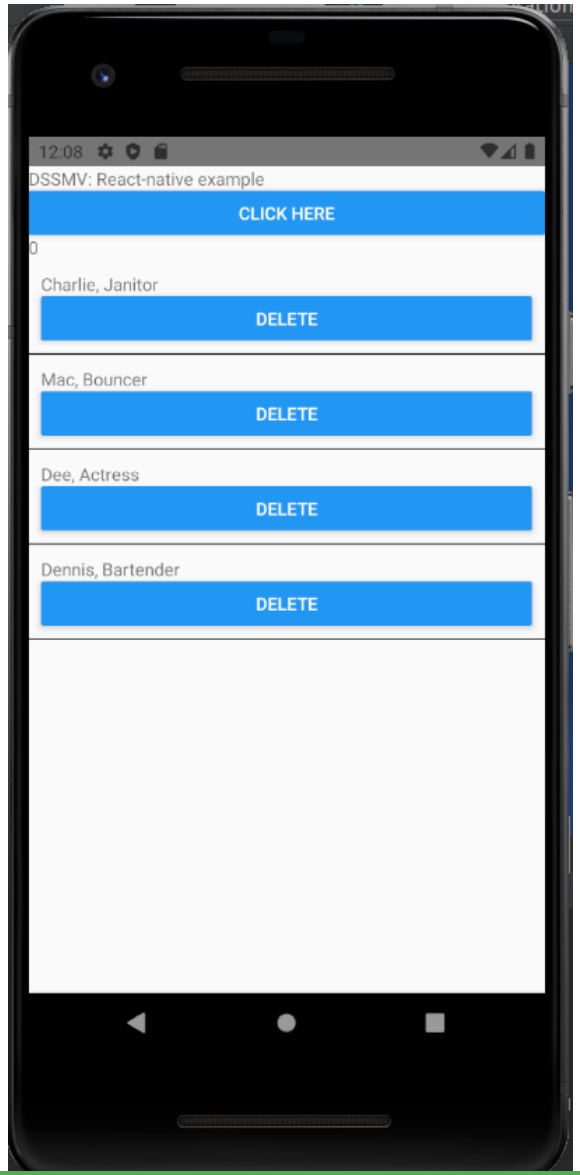
App component (II)

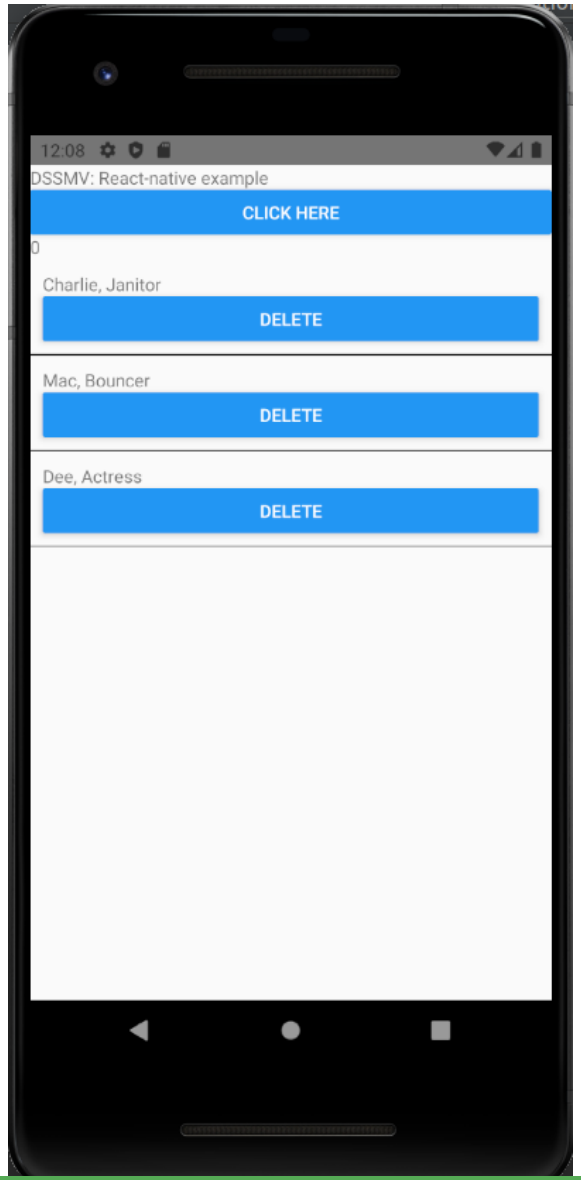
- Changes on App.js file

```
...
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      title: 'DSSMV: React-native example',
      people: data,
    };
  }
  render() {
    const {title, people} = this.state;
    return (
    <View>
      <Text>{title}</Text>
      <Counter />
      <PeopleList listOfItems={people} />
    </View>
    );
  }
}
export default App;
```

FirstApp Example: Fourth iteration

Output


12:08
DSSMV: React-native example
CLICK HERE
0
Charlie, Janitor
DELETE
Mac, Bouncer
DELETE
Dee, Actress
DELETE
Dennis, Bartender
DELETE


12:08
DSSMV: React-native example
CLICK HERE
0
Charlie, Janitor
DELETE
Mac, Bouncer
DELETE
Dee, Actress
DELETE

PeopleListItem component

- Changes on PeopleListItem.js file

```
import React, {Component} from 'react';
import {View, Text, Button} from 'react-native';

class PeopleListItem extends Component {
  constructor(props) {
    super(props);
  }
  handleClick = name => {
    this.props.handleClick(name);
  };
  render() {
    const {name, job} = this.props;
    return (
    <View
      style={{borderBottomColor: 'black', borderBottomWidth: 1, padding: 10}}>
      <Text>
      {name}, {job}
      </Text>
      <Button onPress={() => this.handleClick(name)} title="Delete" />
    </View>
    );
  }
}
export default PeopleListItem;
```

PeopleList component

- Changes on PeopleList.js file

```
...
class PeopleList extends Component {
  constructor(props) {
    super(props);
  }
  handleClick = name => {
    this.props.deleteItem(name);
  };
  render() {
    const people = this.props.listOfItems;
    return (
    <FlatList
    data={people}
    keyExtractor={item => item.id}
    renderItem={({item}) => (
      <PeopleListItem key={item.id} name={item.name} job={item.job} handleClick={this.
          handleClick}/>
    )}
    />
    );
  }
}
export default PeopleList;
```

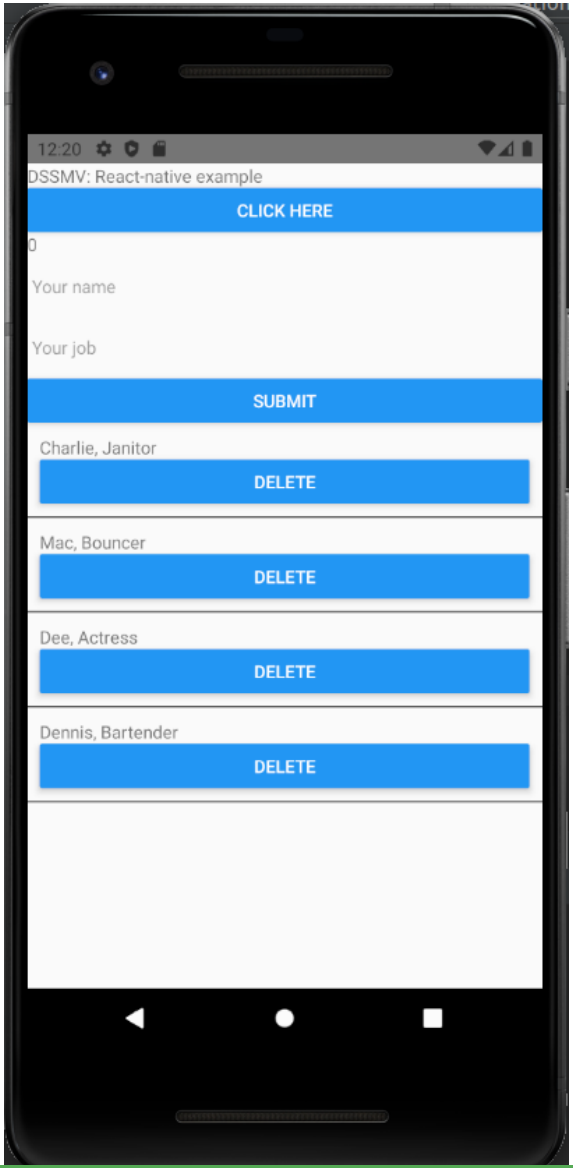
App component

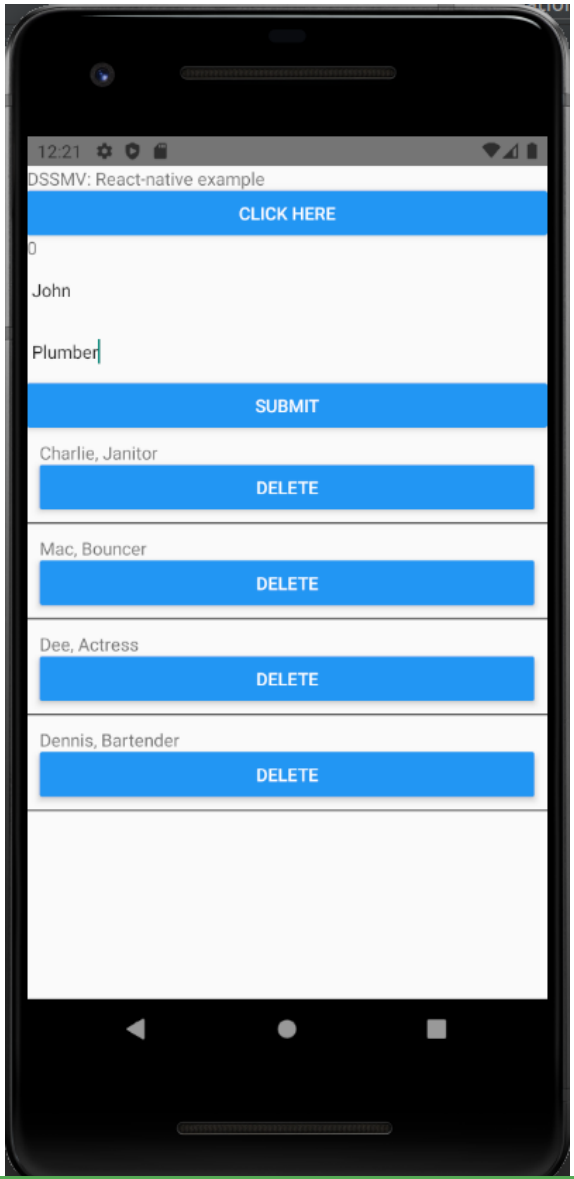
- Changes on App.js file

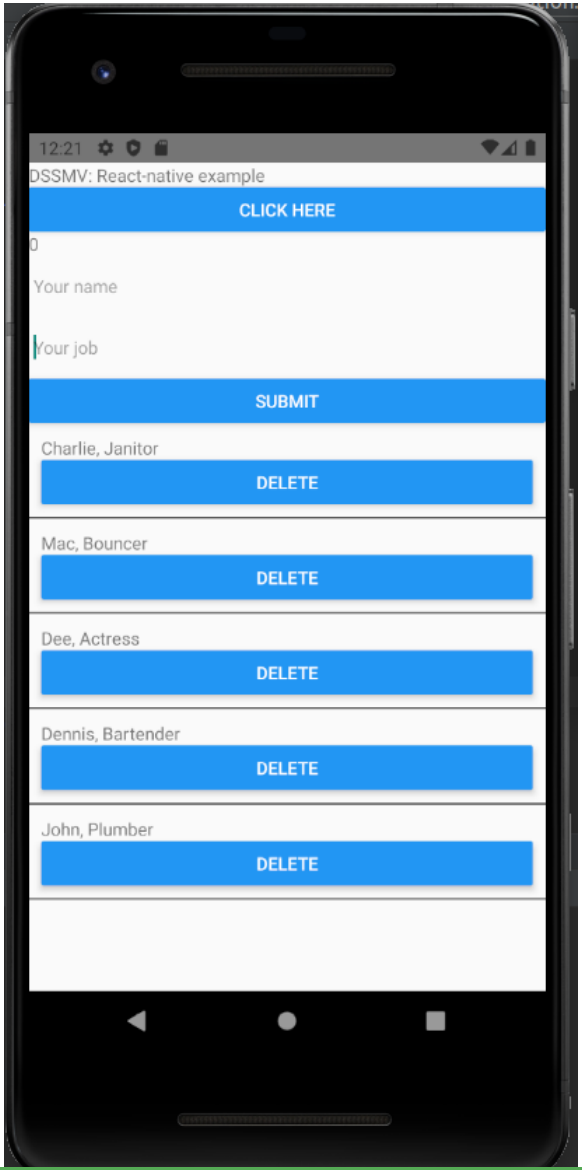
```
...
class App extends Component {
  constructor(props) {
    ...
  }
  deletePerson = name => {
    const {people} = this.state;
    let newList = [...people];
    newList = newList.filter(item => item.name.trim() !== name.trim());
    this.setState({people: newList, });
  };
  render() {
    const {title, people} = this.state;
    return (
    <View>
      <Text>{title}</Text>
      <Counter />
      <PeopleList listOfItems={people} deleteItem={this.deletePerson} />
    </View>
    );
  }
}
export default App;
```

FirstApp Example: Fifth iteration

Output


12:20
DSSMV: React-native example
CLICK HERE
0
Your name
Your job
SUBMIT
Charlie, Janitor
DELETE
Mac, Bouncer
DELETE
Dee, Actress
DELETE
Dennis, Bartender
DELETE


12:21
DSSMV: React-native example
CLICK HERE
0
John
Plumber
SUBMIT
Charlie, Janitor
DELETE
Mac, Bouncer
DELETE
Dee, Actress
DELETE
Dennis, Bartender
DELETE


12:21
DSSMV: React-native example
CLICK HERE
0
Your name
Your job
SUBMIT
Charlie, Janitor
DELETE
Mac, Bouncer
DELETE
Dee, Actress
DELETE
Dennis, Bartender
DELETE
John, Plumber
DELETE

Form component (I)

- Create Form.js file.

```
import React, {Component} from 'react';
import {View, Button, TextInput, Alert} from 'react-native';
const initialState = {name: '', job: ''};
class Form extends Component {
  constructor(props) {
    super(props);
    this.state = initialState;
  }
  onChangeNameText = text => {
    this.setState({name: text});
  };
  onChangeJobText = text => {
    this.setState({job: text});
  };
  submitForm = () => {
    const {name, job} = this.state;
    if (name.length > 0 && job.length > 0) {
      this.props.addItem(this.state);
      this.setState(initialState);
    } else {
      Alert.alert('Please enter data!!');
    }
  };
  ...
```

Form component (II)

```
...
  render() {
    const {name, job} = this.state;
    return (
    <View>
      <TextInput
        placeholder="Your name"
        maxLength={20}
        onChangeText={this.onChangeNameText}
        value={name}
      />
      <TextInput
        placeholder="Your job"
        maxLength={20}
        onChangeText={this.onChangeJobText}
        value={job}
      />
      <Button onPress={this.submitForm} title="Submit" />
    </View>
    );
  }
}
export default Form;
```

App component

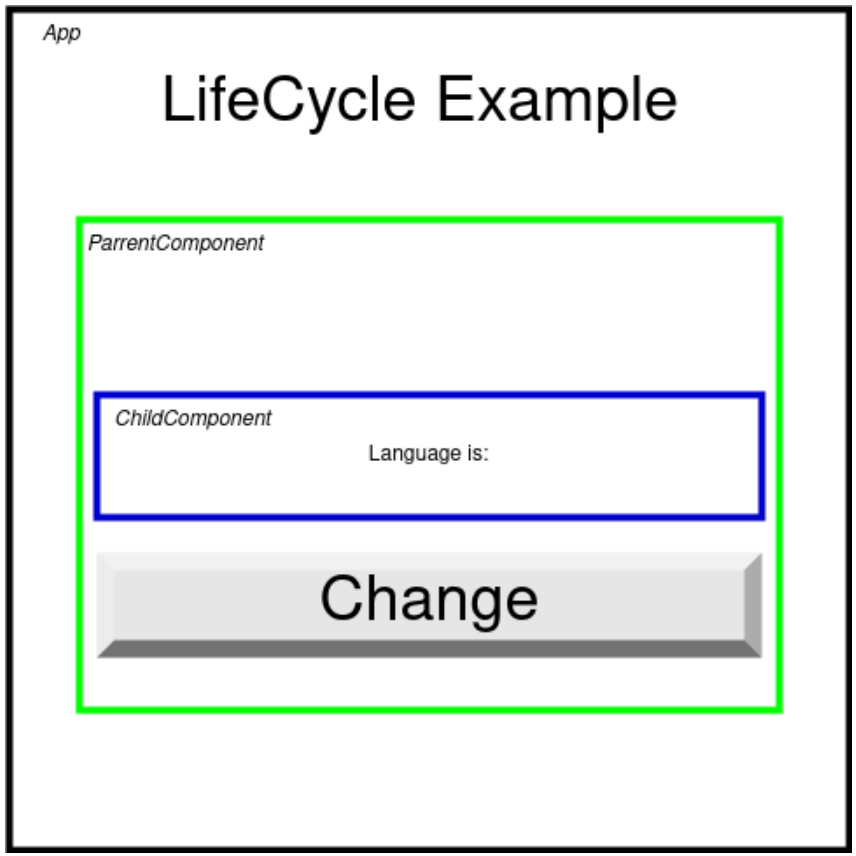
- Changes on App.js file

```
...
import PeopleList from './PeopleList';
import Form from './Form';
const data = [ ... ];
class App extends Component {
  constructor(props) { ... }
  deletePerson = name => { ... };
  addPerson = person => {
    const {people} = this.state;
    let newList = [...people, person];
    this.setState({people: newList});
  };
  render() {
    const {title, people} = this.state;
    return (
    <View>
      <Text>{title}</Text>
      <Counter />
      <Form addItem={this.addPerson} />
      <PeopleList listOfItems={people} deleteItem={this.deletePerson} />
    </View>
    );
...
```

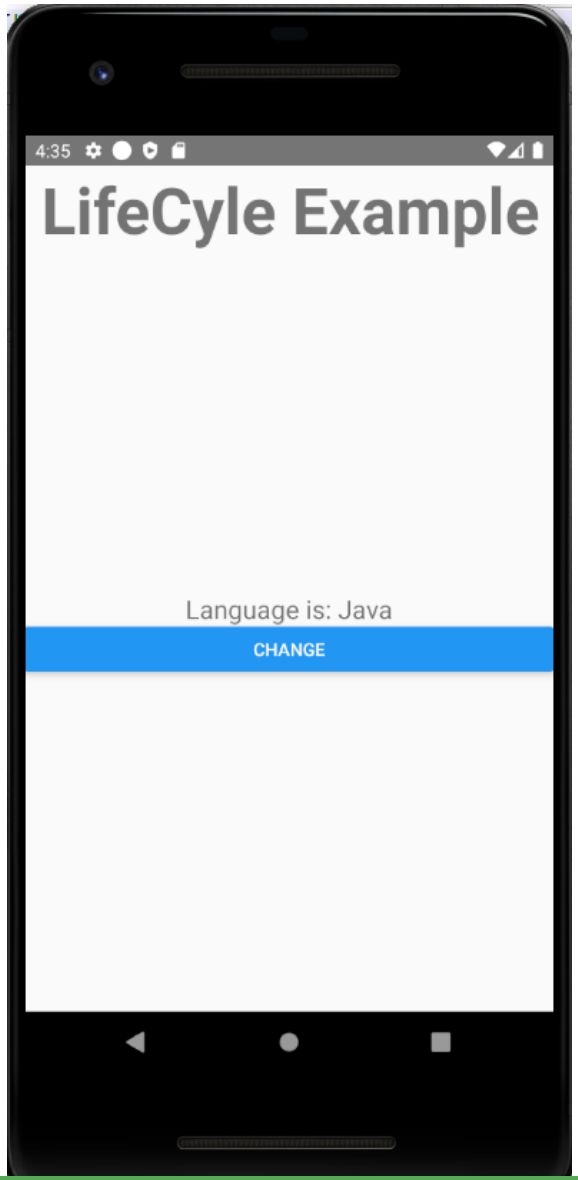
Lifecycle

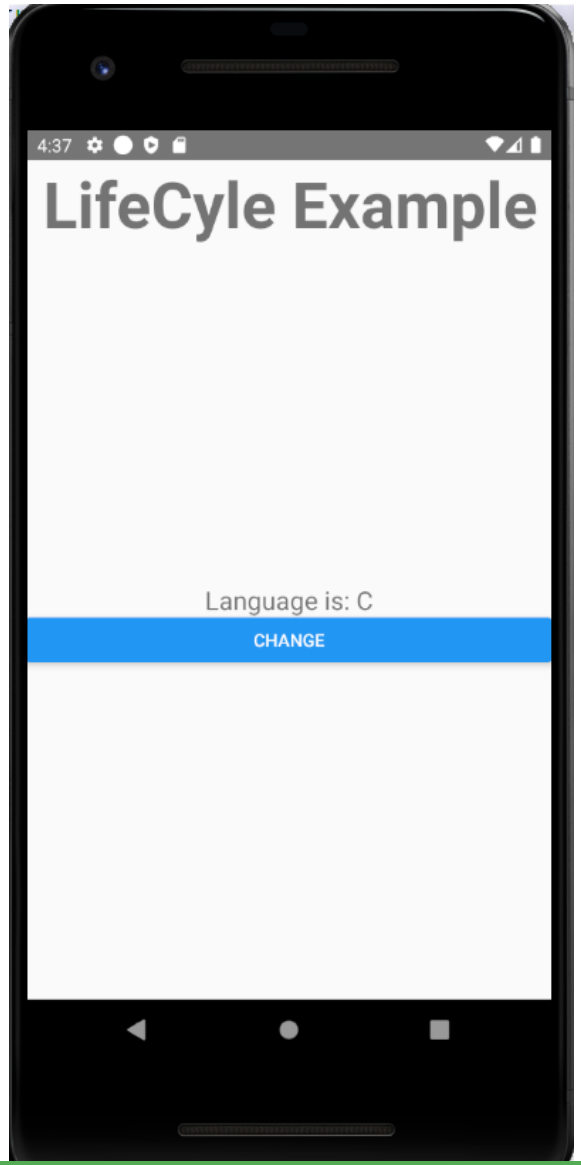
`App` component

- Create an app called `Lifecycle`
- It is composed by three components: `App`, `ParentComponent`, and `ChildComponent`



Output


4:35
LifeCyle Example
Language is: Java
CHANGE


4:37
LifeCyle Example
Language is: C
CHANGE

`ChildComponent` component (I)

- Create `ChildComponent.js` file.

```
import React, {Component} from 'react';
import {Text, View} from 'react-native';
class ChildComponent extends Component {
  constructor(props) {
    super(props);
    console.log('ChildComponent: Constructor called.');
  }
  componentDidMount() {
    console.log('ChildComponent: componentDidMount called.');
  }
  componentDidUpdate(prevProp, prevState) {
    console.log('ChildComponent: componentDidUpdate called.');
  }
  componentWillUnmount() {
    console.log('ChildComponent: componentWillUnmount called.');
  }
...
```

`ChildComponent` component (II)

```
...

  render() {
    console.log('ChildComponent: render called.');
    return (
    <View style={{justifyContent: 'center', alignItems: 'center'}}>
    <Text style={{fontSize: 20}}>Language is: {this.props.name}</Text>
    </View>
    );
  }
}
export default ChildComponent;
```

`ParentComponent` component (I)

- Create `ParentComponent.js` file.

```
import React, {Component} from 'react';
import {View, Button} from 'react-native';
import ChildComponent from './ChildComponent';

class ParentComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      language: 'Java',
    };
    console.log('ParentComponent: Constructor called.');
  }
  handleClick = () => {
    console.log('App: handleClick called.');
    let {language} = this.state;
    language = language == 'Java' ? 'C' : 'Java';
    this.setState({language: language});
  };
  componentDidMount() {
    console.log('ParentComponent: componentDidMount called.');
  }

...
```

`ParentComponent` component (II)

```
...
  componentDidUpdate(prevProp, prevState) {
    console.log('ParentComponent: componentDidUpdate called.');
  }

  componentWillUnmount() {
    console.log('ParentComponent: componentWillUnmount called.');
  }
  render() {
    console.log('ParentComponent: render called.');
    const {language} = this.state;
    return (
    <View style={{flex: 1, justifyContent: 'center'}}>
    <ChildComponent name={language} />
    <Button onPress={() => this.handleClick()} title="Change" />
    </View>
    );
  }
}

export default ParentComponent;
```

App component (I)

- Change `App.js` file.

```
import React, {Component} from 'react';
import {View, Text} from 'react-native';
import ParentComponent from './ParentComponent';

class App extends Component {
  componentDidMount() {
    console.log('App: componentDidMount called.');
  }

  componentDidUpdate(prevProp, prevState) {
    console.log('App: componentDidUpdate called.');
  }

  componentWillUnmount() {
    console.log('App: componentWillUnmount called.');
  }
  ...
```

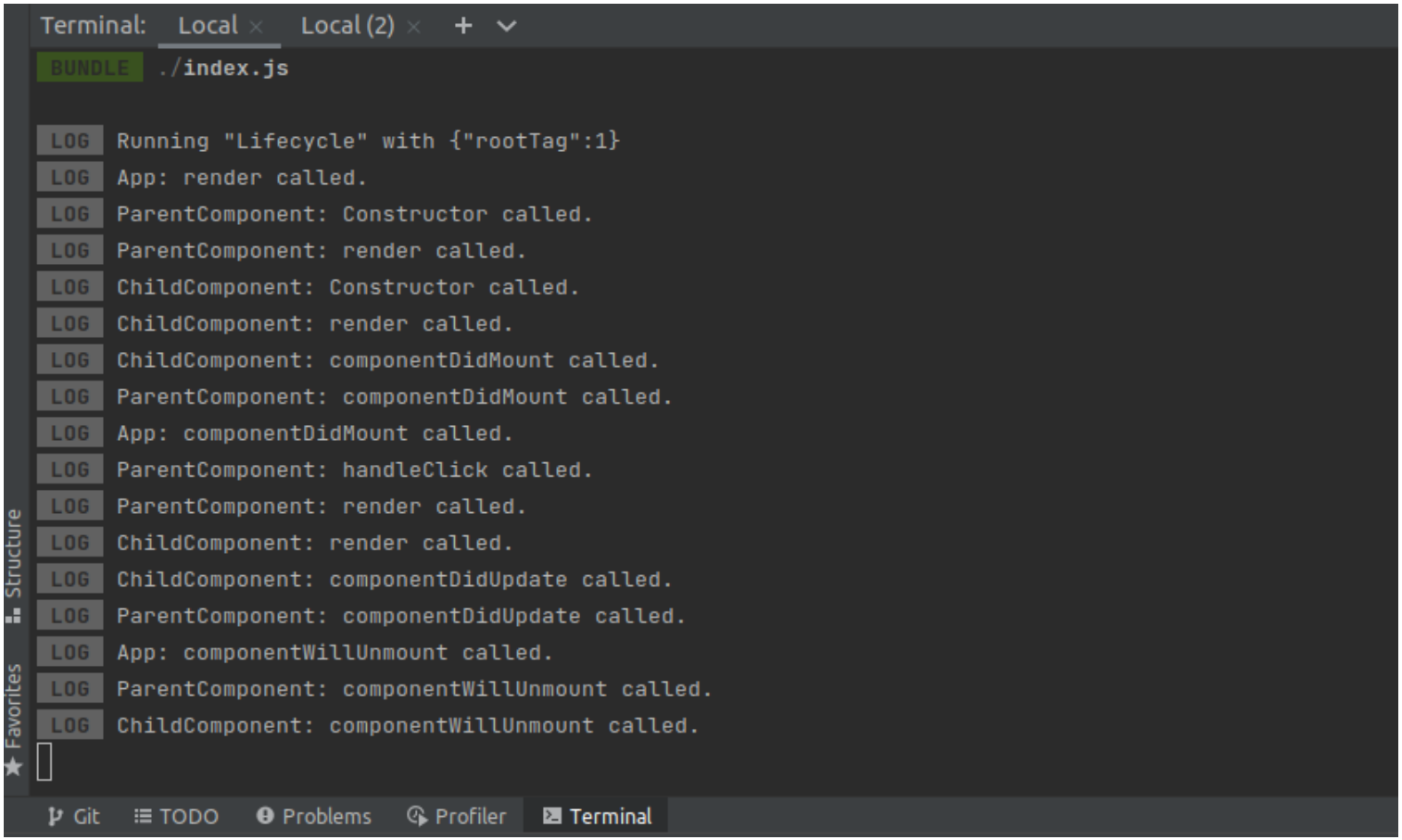
App component (II)

```
...
   render() {
     console.log('App: render called.');
     return (
     <View style={{flex: 1, justifyContent: 'center'}}>
     <Text style={{fontSize: 50, fontWeight: 'bold'}}> LifeCyle Example</Text>
     <ParentComponent/>
     </View>
     );
   }
 }

 export default App;
```

Log messages

- Log messages appear in the running Metro terminal.



Bibliography

Resources

- David Flanagan, "JavaScript: The Definitive Guide", O'Reilly Media, Inc., 2020
- Adam Boduch and Roy Derks, "React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd Edition",Packt Publishing, 2020
- Dan Ward, "React Native Cookbook Second Edition Step-by-step recipes for solving common React Native development problems", ,Packt Publishing, 2019
- `https://reactnative.dev/`
- `https://reactjs.org/`
- `https://reactnavigation.org/`