Universidad Nacional de Quilmes

Construcción de Interfaces de Usuario

Departamento de Ciencia y Tecnología

Nombre y Apellido:

Parcial

22/11/2018

Ejercicio 1: Desktop

Dado el siguiente código Xtend:

```
class FileSearchWindow extends MainWindow<FileSearch> {
  new() { super(new FileSearch) }
  override void createContents(Panel mainPanel) {
    this.title = "Buscador de archivos"
    mainPanel.layout = new VerticalLayout
    new Panel(mainPanel) => [
      layout = new HorizontalLayout
      new Selector<String>(it) => [
        allowNull(false); items <=> "directories"; value <=> "currentDirectory"
      ]
      new TextBox(it) => [ value <=> "search"; width = 300 ]
      new Button(it) => [ caption = "Buscar"; setAsDefault; onClick[| search ] ]
    ]
    new Panel(mainPanel) => [
      val files = new Table<File>(it, typeof(File)) => [
       items <=> "foundFiles"; numberVisibleRows = 5; value <=> "selectedFile"
      ]
      new Column<File>(files) => [
       title = "Nombre del archivo"; fixedSize = 150; bindContentsToProperty("name")
      ]
      new Column<File>(files) => [
       title = "Tamaño"; fixedSize = 50; bindContentsToProperty("formattedSize")
      ]
      new Column<File>(files) => [
       title = "Modificado"; fixedSize = 50; bindContentsToProperty("formattedUpdated")
      ]
      new Column<File>(files) => [
       title = "Ubicación"; fixedSize = 250; bindContentsToProperty("path")
      ]]}
  def search() { modelObject.searchFiles }
  def static main(String[] args) { new FileSearchWindow().startApplication }
} // Fin FileSearchWindow

@Accessors
@Observable
class FileSearch {
  String f; List<String> directories; String currentDirectory
  List<File> files; List<File> foundFiles; File selectedFile
    new() {
    search = ""; currentDirectory = '/home/ui'
    directories = #['/', '/etc', '/home/ui', '/var', '/home/ui/code'].sort
    files = #[
```

```
      newFile("index.html", 41650, "/etc", "10/09/2019"),
      newFile("Home.jsx", 19380, "/home/ui", "05/10/2017"),
      newFile("package.json", 94830, "/home/ui/code", "28/08/2017"),
      newFile("pasaron-cosas.txt", 43670, "/etc", "06/01/2019"),
      newFile("apuntes.md", 83810, "/var", "18/01/2019"),
      newFile("Logo.png", 48690, "/home/ui/code", "28/10/2019"),
      newFile("phpunit.xml", 11970, "/", "22/12/2018"),
      newFile("express.js", 12890, "/var", "19/09/2017"),
      newFile("robots.txt", 16390, "/home/ui", "16/08/2017"),
      newFile("favicon.ico", 76210, "/", "22/12/2017"),
      newFile("meme.jpg", 88950, "/etc", "18/08/2017"),
      newFile("app.js", 78590, "/etc", "08/11/2018"),
      newFile("pom.xml", 22650, "/etc", "26/01/2019"),
      newFile("Movie.jsx", 17390, "/var", "19/04/2018")
    ]
    foundFiles = files.take(5).clone
  }
  def newFile(String name, int size, String path, String date) {
    val formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy")
    new File(name, size, path, LocalDate.parse(date, formatter))
  }
  def searchFiles() { /* TODO */ }
} // Fin FileSearch

@Accessors
@Observable
class File {
  String name; int size; String path; LocalDate updated
  new(String name, int size, String path, LocalDate updated) {
    this.name = name; this.size = size; this.path = path; this.updated = updated
  }
  def formattedUpdated() {
    val formatter = DateTimeFormatter.ofPattern("dd.MM.yyyy");
    updated.format(formatter);
  }
  def formattedSize() '''«Math.round(size / 1000)» KB'''
} // Fin File
```

Responder:

1. Identifique justificadamente los objetos de vista y los de modelo
2. ¿Qué tipo de ventana es la definida en el código? ¿Existen otros tipos de ventana? ¿Cuáles y en qué se diferencian?
3. Dibujar la ventana de Arena que resulta de ejecutar el programa. La ventana debe dibujarse en su estado inicial, o sea, sin haber interactuado con la misma. Tener en cuenta si debe o no mostrarse información del modelo.
4. Explicar, con sus palabras, el concepto de binding e identificar todos los casos que aparecen en el código.
5. Determinar si, con el código planteado, existe algún tipo de búsqueda en funcionamiento. De no existir, implementar el código necesario para que funcione. Caso contrario explicar por qué funciona.

Ejercicio 2: Web

Se está desarrollando una aplicación web para mantener notas personales, al estilo *Google Keep*. Por el momento se cuenta con un servicio REST que expone información de las notas y un front-end en ReactJS que se encarga del renderizado.

Por el momento el servicio REST expone las notas con una estructura como la que se observa a continuación:

```
{
 "notes": [
  {"id":1, "color":"Coral", "contents":"Diseño de Objetos con TDD" },
  {"id":2, "color":"CornflowerBlue", "contents":"Xtend, es como Java pero chévere" },
  {"id":3, "color":"GoldenRod", "contents":"Diseñando Interfaces con Arena" },
  {"id":4, "color":"LimeGreen", "contents":"Transformers & Adapters" },
  {"id":5, "color":"Coral", "contents":"Mundo Web: HTTP, HTML, CSS" },
  {"id":6, "color":"MediumPurple", "contents":"API Rest: JSON por doquier" },
  {"id":7, "color":"Orchid", "contents":"Componentes con ReactJS" },
  {"id":8, "color":"CornflowerBlue", "contents":"¿Algo de Mobile?" }
 ]
}
```

Y la aplicación de ReactJS corriendo en http://localhost:3000 con el siguiente código:

```
class App extends Component {
  render() { return (
      <BrowserRouter><Switch>
        <Route path="/data" render={props => <Data {...props} />} />
        <Route path="/data/table" render={props => <DataTable {...props} />} />
        <Route path="/info" render={props => <Info {...props} />} />
        <Route path="/error" render={props => <Error {...props} />} />
        <Route path="/" render={props => <Home {...props} />} />
      </Switch></BrowserRouter>
  );}
}
const Home = () => <h1>Home Page</h1>;
const Error = () => <h1>Error Page</h1>;
const Data = props => props.history.push('/error');
const DataTable = props => props.history.push('/info');
class Info extends Component {
  constructor(props) { super(props); this.state = { notes: [] }; }
  componentDidMount() {
    axios.get('http://localhost:3003/notes').then((response) => {
      this.setState({ notes: response.data });
    });
  }
  rowsWith(notes, i) {
    return <div className="row" key={i}>{notes.map(note => this.col(note))}</div>;
  }
  col(note) {
    const styles = { margin: '5px', padding: '5px',
        border: '1px solid grey', borderRadius: '7px' };
    return <div key={note.key} className="col" style={styles}>{note.contents}</div>;
  }
```

```
  renderNotes() {
    const cols = 3; const rows = []; const amount = this.state.notes.length;
    for (let i = 0; i < amount; i += cols) {
      rows.push(this.rowsWith(this.state.notes.slice(i, i + cols), i));
    }
    return rows;
  }
  render() { return (
    <React.Fragment><div className="container">
      <h1>Info UI Page</h1>
      {this.renderNotes()}
    </div></React.Fragment>
  );}
}
export default App;
```

Responder:

1. Si se ingresa a http://localhost:3000/data/table, ¿qué componente que se renderiza? ¿Existe algún bug? Si existe, ¿cómo lo arreglaría? Si no existe, explique por qué.
2. El componente *Info* es el encargado de renderizar las notas obtenidas de la API, ¿Funciona correctamente? Si funciona correctamente, explique cómo funciona el workflow de ReactJS en este componente. Si no funciona, explique cómo lo arreglaría.
3. Asumiendo que el componente *Info* funciona (porque ya funcionaba o porque fue arreglado), dibuje el renderizado HTML resultante de la ejecución del componente, teniendo en cuenta la información disponible del servicio REST planteado.
4. Las notas provista por el servicio REST incluyen un color. Modifique el código necesario del componente *Info* para que cada nota tenga como color de fondo el color provisto por la API (no hace falta que reescriba todo el componente, solo aquellas partes que es necesario que cambien).
5. Se quiere agregar una botonera con las funciones “Compartir”, “Editar” y “Borrar”. Modifique el componente agregando el código HTML necesario para que aparezca la botonera al pie de cada nota. No hace falta que escriba el comportamiento javascript, simplemente el maquetado. Si quiere utilizar iconografía puede hacerlo con nombres inventados de íconos.