Universidad Nacional de Quilmes

Construcción de Interfaces de Usuario

Departamento de Ciencia y Tecnología

Nombre y Apellido:

Parcial

27/06/2019

Ejercicio 1: Desktop

Dado código Arena mostrado a continuación (puede asumir imports):

1. **(3,5)** Dibuje la/las ventanas que se describen en el flujo del programa (analice todas las posibilidades).
2. **(0,5)** ¿Qué particularidad tiene la opción C?
3. **(1,0)** Defina el concepto de Binding con sus palabras e identifique todos los casos de binding planteados en el código.

```
fun main() = LuckyWindow(Lucky()).startApplication()

class LuckyWindow(lucky: Lucky) : MainWindow<Lucky>(lucky) {
  override fun createContents(panel: Panel) {
    this.title = "Voy a tener suerte"
    Label(panel)
      .setBackground(WHITE).setForeground(RED)
      .bindValueToProperty<Any, ControlBuilder>("message")

    Label(panel).setText("Elige una opción")

    val container = Panel(panel).setLayout(HorizontalLayout())
    val options = RadioSelector<Any>(container)
    options.bindItemsToProperty("options")
    options.bindValueToProperty<Any, ControlBuilder>("selected")

    Button(container).setCaption("Voy a Tener Suerte")
      .onClick { openFrom(modelObject.selected) }
  }
```

```
private fun openFrom(option: String) {
    when (option) {
      "A" -> LuckyDialog(this, modelObject)
      "B" -> modelObject.message = "Esto está roto"
      "C" -> {
        this.close()
        LuckyDialog(this, modelObject).open()
      }
      else -> modelObject.message = "Tenés que seleccionar algo"
    }
  }
}

class LuckyDialog(owner: WindowOwner, model: Lucky) : Dialog<Lucky>(owner, model) {
  override fun createFormPanel(panel: Panel) {
    title = "¿Tuviste suerte?"
    Label(panel).setText(modelObject.lucky)
    Button(panel).setCaption("OK").onClick { this.close() }
  }
}

@Observable
class Lucky {
  val options = listOf("A", "B", "C")
  var lucky = ""
  var message = ""
  var selected = ""
      set(value) {
        field = value
        when (value) {
          "A" -> lucky = "Perdiste!"
          "C" -> lucky = "Ganaste!"
        }
      }
}
```

Ejercicio 2: API / Web

Dado los siguientes códigos de Javalin y React (puede asumir imports):

1. **(1,0)** Describa todas las rutas definidas por la API, incluyendo métodos HTTP.
2. **(0,5)** Asumiendo que la aplicación web está corriendo en `localhost:3000`, y la API en `localhost:6001`, describa qué sucede y qué se renderiza si se ingresa por browser a `http://localhost:3000/`.
3. **(3,0)** Dibuje el renderizado de la página web luego de ingresar a `http://localhost:3000/todas`.
4. **(0,5)** Describa con sus palabras qué tipos de lenguajes son HTML, CSS y JS, en qué se diferencian y cómo se complementan.

```
// API :: Javalin
fun main() {
  val app = Javalin.create().enableCorsForAllOrigins().enableRouteOverview("rutas").start(6001)
  val controller = Controller()
  app.routes {
    delete("/") { ctx -> ctx.json("Ups, pasaron cosas!") }
    path("notas") {
      get { ctx -> ctx.status(200); ctx.json(controller.getNotes()) }
      post { ctx ->
        controller.addNote(ctx.body<Map<String, Any>>())
        ctx.status(201); ctx.json(controller.lastNote())
      }
    }
  }
}

class Controller {
  private var lastIndex = 0
  private val notes = mutableListOf<Map<String, Any>>()
  fun getNotes() = notes
  fun lastNote() = notes.last()
  fun addNote(note: Map<String, Any>) {
    if ("name" !in note) throw BadRequestResponse("Mandaste cualquiera")
    lastIndex++
    notes.add(mapOf("id" to lastIndex, "name" to note["name"].toString()))
  }
}

// Notas cargadas en la API
[
 {"id": 1, "name": "Si no leo todo el enunciado hay tabla"},
 {"id": 2, "name": "Acordate de releer lo que escribiste antes de entregar"},
 {"id": 3, "name": "Sacale los pirinchos a las hojas que arrancaste del cuaderno"}
]
```

```
// Web :: React
const server = 'http://localhost:6001';
const request = (type, path, body) => axios
  .request({ url: `${server}${path}`, method: type, data: body })
  .then(req => req.data);

export default class App extends React.Component {
  render() {
    return (
      <BrowserRouter>
        <Switch>
          <Route path="/todas" component={Todas} />
          <Route path="/error/:message" render={Error} />
          <Route path="/" render={Home} />
        </Switch>
      </BrowserRouter>
    );
  }
}

const Home = props => props.history.push('/error/Todo%20Mal');
const Error = props => <h1>Error Page {props.match.params.message}</h1>;

class Todas extends React.Component {
  constructor(props) { super(props); this.state = { notes: [] } }
  componentDidMount() { request('get', '/notas').then(notes => this.setState({ notes })) }
  render() {
    return (<div>
      <h1>Listado de Notas</h1>
       <div>{this.state.notes.map(note => <Nota key={note.id} id={note.id} name={note.name} />)}</div>
    </div>);
  }
}

const Nota = props => <div style={{ border: '1px solid blue', padding: '10px', margin: '10px' }}>
  <div style={{ borderBottom: '1px dashed black', marginBottom: '5px' }}>#{props.id}</div>
  <div>{props.name}</div>
</div>
```