IISSI2 REACT-NATIVE COMPONENTS CHEAT SHEET

0. View

```
<View style={styles.container}>
        ...
</View>

return (
  <>
    <View>
      <Text>Some text</Text>
    </View>
    <View>
      <Text>Some other text</Text>
    </View>
  </>
)

<ScrollView> ... </ScrollView>
```

1. Text

```
<Text> Some text </Text>


<TextRegular style = {{fontSize: 16, alignSelf: 'center', margin: 20}}>
        Escribe aquí.
</TextRegular>


<TextRegular style={styles.text}>
        {restaurant.description}
</TextRegular>
```

2. Pressable / Button

```
<Pressable

        onPress = {() => {
          navigation.navigate('RestaurantDetailScreen', { id:
Math.floor(Math.random() * 100) })
        }}

        Style = {({ pressed }) => [
          {
```

```
            backgroundColor: pressed
              ? GlobalStyles.brandBlueTap
              : GlobalStyles.brandBlue
          },
          styles.actionButton
        ]}
      >

        <TextRegular textStyle={styles.text}>
                 Texto del boton aquí.
        </TextRegular>

</Pressable>

<Pressable
  onPress={() => console.log(`Edit pressed for restaurantId = ${item.id}`)}
  style={({ pressed }) => [
    {
      backgroundColor: pressed
        ? GlobalStyles.brandBlueTap
        : GlobalStyles.brandBlue
    },
    styles.actionButton
  ]}>
  <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center'
}]}>
    <MaterialCommunityIcons name='pencil' color={'white'} size={20}/>
    <TextRegular textStyle={styles.text}>
      Edit
    </TextRegular>
  </View>
</Pressable>

<Pressable
  onPress={() => console.log(`Delete pressed for restaurantId = ${item.id}`)}
  style={({ pressed }) => [
    {
      backgroundColor: pressed
        ? GlobalStyles.brandPrimaryTap
        : GlobalStyles.brandPrimary
    },
    styles.actionButton
  ]}>
  <View style={[{ flex: 1, flexDirection: 'row', justifyContent: 'center'
}]}>
    <MaterialCommunityIcons name='delete' color={'white'} size={20}/>
    <TextRegular textStyle={styles.text}>
      Delete
    </TextRegular>
  </View>
</Pressable>
import { MaterialCommunityIcons } from '@expo/vector-icons'

 <TouchableHighlight onPress={() => {
            navigation.navigate('RestaurantDetailScreen', { id:
restaurant.id })
          }}>
```

```
              <Image style={styles.image} source={restaurant.logo ? {
uri: process.env.API_BASE_URL + '/' + restaurant.logo, cache: 'force-
cache' } : defaultRestaurantLogo}/>
            </TouchableHighlight>
```

3. FlatList

```
<FlatList
        style = {styles.container}
        data = {restaurants}
        renderItem = {renderRestaurant}
        keyExtractor = {item => item.id.toString()}
      />
```

```
const renderEmptyProductsList = () => {
    return (
      <TextRegular textStyle={styles.emptyList}>
        This restaurant has no products yet.
      </TextRegular>
    )
  }
const fetchRestaurantDetail = async () => {
    try {
      const fetchedRestaurant = await getDetail(route.params.id)
      setRestaurant(fetchedRestaurant)
    } catch (error) {
      showMessage({
        message: `There was an error while retrieving restaurant details
(id ${route.params.id}). ${error}`,
        type: 'error',
        style: GlobalStyles.flashStyle,
        titleStyle: GlobalStyles.flashTextStyle
      })
    }
  }

  return (
      <FlatList
          ListHeaderComponent={renderHeader}
          ListEmptyComponent={renderEmptyProductsList}
          style={styles.container}
          data={restaurant.products}
          renderItem={renderProduct}
          keyExtractor={item => item.id.toString()}
        />
  )
}
```

4. Images

```
const renderHeader = () => {
    return (

      <ImageBackground source={(restaurant?.heroImage) ? { uri:
process.env.API_BASE_URL + '/' + restaurant.heroImage, cache: 'force-cache' }
: undefined } style={styles.imageBackground}>

        <View style={styles.restaurantHeaderContainer}>

            <TextSemiBold textStyle={styles.textTitle}>
                {restaurant.name}
            </TextSemiBold>

            <Image style={styles.image} source={restaurant.logo ? { uri:
process.env.API_BASE_URL + '/' + restaurant.logo, cache: 'force-cache' } :
undefined} />

            <TextRegular textStyle={styles.text}>
                {restaurant.description}
            </TextRegular>

        </View>

      </ImageBackground>
    )
  }
```

```
<ImageCard

 imageUri={item.logo ? { uri: process.env.API_BASE_URL + '/' + item.logo } : undefined}

 title={item.name}

 onPress={() => navigation.navigate('RestaurantDetailScreen', { id: item.id })

 }}

>

</ImageCard>
```

5. Input Item

```
<InputItem
        name='sampleInput'
        label='Sample input'
      />

<Formik
```

```
    initialValues={initialValues}
    >

      {({ setFieldValue, values }) => (

        <ScrollView>
          <View style={{ alignItems: 'center' }}>
            <View style={{ width: '60%' }}>

              <InputItem
                name='name'
                label='Name:'
              />

              {/* Any other inputs */}

            </View>
          </View>
        </ScrollView>
      )}

    </Formik>
```

Ejemplo de image pickers en laboratorio 6 punto 2.1.2

```
<InputItem
                    name = {item.id.toString()}
                    placeholder = { initialProductsValues[item.id] || '0'
}
                    value = { initialProductsValues[item.id] || '0' }
                    onChange={(newQuantity) => handleQuantityChange(item,
newQuantity)}
                  />
```

6. Drop Down Picker

import DropDownPicker from 'react-native-dropdown-picker'

const [restaurantCategories, setRestaurantCategories] = useState([])

const [open, setOpen] = useState(false)

```
useEffect(() => {
 async function fetchRestaurantCategories () {
  try {
   const fetchedRestaurantCategories = await getRestaurantCategories()
   const fetchedRestaurantCategoriesReshaped = fetchedRestaurantCategories.map((e) => {
    return {
     label: e.name,
     value: e.id
```

```
      }
    })
    setRestaurantCategories(fetchedRestaurantCategoriesReshaped)
   } catch (error) {
    showMessage({
     message: `There was an error while retrieving restaurant categories. ${error} `,
     type: 'error',
     style: GlobalStyles.flashStyle,
     titleStyle: GlobalStyles.flashTextStyle
    })
   }
  }
  fetchRestaurantCategories()
}, [])
```

```
<DropDownPicker
  open={open}
  value={values.restaurantCategoryId}
  items={restaurantCategories}
  setOpen={setOpen}
  onSelectItem={ item => {
    setFieldValue('restaurantCategoryId', item.value)
  }}
  setItems={setRestaurantCategories}
  placeholder="Select the restaurant category"
  containerStyle={{ height: 40, marginTop: 20 }}
  style={{ backgroundColor: GlobalStyles.brandBackground }}
  dropDownStyle={{ backgroundColor: '#fafafa' }}
/>
```

7. SWITCH

```
<TextRegular style={styles.switch}>Is it available?</TextRegular>

<Switch
  trackColor={{ false: GlobalStyles.brandSecondary, true:
GlobalStyles.brandPrimary }}
  thumbColor={values.availability ? GlobalStyles.brandSecondary : '#f4f3f4'}
  value={values.availability}
  style={styles.switch}
  onValueChange={value =>
    setFieldValue('availability', value)
  }
/>
```

8. POST VALIDATION SCHEMA

```
import { ErrorMessage, Formik } from 'formik'

import * as yup from 'yup'

const initialRestaurantValues = { name: null, description: null, address: null, postalCode: null, url: null, shippingCosts: null, email: null, phone: null, restaurantCategoryId: null }


const validationSchema = yup.object().shape({

name: yup

  .string()

  .max(255, 'Name too long')

  .required('Name is required'),

…

restaurantCategoryId: yup

  .number()

  .positive()

  .integer()

  .required('Restaurant category is required')

})

<Formik

  validationSchema={validationSchema}

  initialValues={initialRestaurantValues}

  onSubmit={createRestaurant}>

  {({ handleSubmit, setFieldValue, values }) => (

    <ScrollView>

      /* Your views, form inputs, submit button/pressable */

    </ScrollView>

  )}

</Formik>


const createRestaurant = async (values) => {
```

```
  setBackendErrors([])
  try {
    const createdRestaurant = await create(values)
    showMessage({
      message: `Restaurant ${createdRestaurant.name} succesfully created`,
      type: 'success',
      style: GlobalStyles.flashStyle,
      titleStyle: GlobalStyles.flashTextStyle
    })
    navigation.navigate('RestaurantsScreen', { dirty: true })
  } catch (error) {
    console.log(error)
    setBackendErrors(error.errors)
  }
}
```

```
const initializeValues = () => {
    restaurant.products.forEach(product => {
      initialProductsValues[product.id] = 0
      validationData[product.id] = yup.number('It must be a
number.').min(0, 'It must be 0 or positive.').integer('It must be
integer.').max(2000, 'The amount is too big.')
    })
    setValidationSchema(yup.object().shape(validationData))
  }
```

9. Modals

```
import DeleteModal from '../../components/DeleteModal'
```

```
import ConfirmationModal from '../../components/ConfirmationModal'
```

```
<DeleteModal
  isVisible={restaurantToBeDeleted !== null}
```

onCancel={() => setRestaurantToBeDeleted(null)}

onConfirm={() => removeRestaurant(restaurantToBeDeleted)}>

<TextRegular>The products of this restaurant will be deleted as well</TextRegular>

<TextRegular>If the restaurant has orders, it cannot be deleted.</TextRegular>

</DeleteModal>

```
<ConfirmationModal
        isVisible={restaurantToBePromoted !== null}
        onCancel={() => setRestaurantToBePromoted(null)}
        onConfirm={() => promoteRestaurant(restaurantToBePromoted)}>
          <TextRegular>The restaurant will be promoted</TextRegular>
          <TextRegular>Are you sure?</TextRegular>
      </ConfirmationModal>
```

```
  const [modalVisible, setModalVisible] = useState(false)
...
<Pressable
          onPress= {() => {
            if (loggedInUser) {
              navigation.navigate('CreateOrderScreen', { id:
restaurant.id })
            } else {
              setModalVisible(true)
            }
          }}
          style={({ pressed }) => [
            {
              backgroundColor: pressed
                ? GlobalStyles.brandPrimary
                : GlobalStyles.brandPrimaryTap
            },
            styles.button
          ]}>
         <View style={[{ flex: 1, flexDirection: 'row', justifyContent:
'center' }]}>
            <MaterialCommunityIcons name='plus-circle' color={'white'}
size={20} />
            <TextRegular textStyle={styles.text}>
              Create Order
            </TextRegular>
          </View>
        </Pressable>
...
<Modal
```

```
        visible={modalVisible}
        animationType="fade"
        transparent={true}
        >
        <View style={{ flex: 1, justifyContent: 'center', alignItems:
'center', backgroundColor: '#00000020' }}>
          <View style={{ backgroundColor: 'white', padding: 25,
borderRadius: 5 }}>
            <TextRegular style={{ fontWeight: 'bold', marginBottom: 20
}}>Please Log In</TextRegular>
            <TextRegular style={{ textAlign: 'center' }}>Please log in to
create an order.</TextRegular>
            <View style={{ flexDirection: 'row', justifyContent: 'space-
between' }}>
            <Pressable onPress={() => setModalVisible(false)} style={{
marginTop: 20 }}>
              <TextRegular style={{ color: 'red', textDecorationLine:
'underline' }}>Close</TextRegular>
            </Pressable>
            </View>
          </View>
        </View>
      </Modal>
```

```
<DeleteModal
        isVisible={orderToBeDeleted !== null}
        onCancel={() => setOrderToBeDeleted(null)}
        onConfirm={() => removeOrder(orderToBeDeleted)}>
          <TextRegular>The order is going to be erased</TextRegular>
      </DeleteModal>
```

10. IMAGE CARD

```
const renderPopularProduct = ({ item }) => {
    return (
      <ImageCard
        imageUri={item.image ? { uri: process.env.API_BASE_URL + '/' +
item.image } : defaultProductImage}
        title={item.name + '    '}
        onPress={() => {
          navigation.navigate('RestaurantDetailScreen', { id:
item.restaurantId })
        }}
```

```
      >
        <TextRegular textStyle={styles.description}
numberOfLines={2}>{item.description} </TextRegular>
        <TextSemiBold
textStyle={styles.price}>{item.price.toFixed(2)}€</TextSemiBold>
        {!item.availability &&
          <TextRegular textStyle={styles.availability }>Not
available</TextRegular>
        }
      </ImageCard>
    )
  }
```

EXTRA

```
const { loggedInUser } = useContext(AuthorizationContext)

useEffect(async () => {
   const fetchedRestaurants = await getAll()
   setRestaurants(fetchedRestaurants)
  }, [])
```

```
import { get } from './helpers/ApiRequestsHelper'

function getAll () {

 return get('/users/myrestaurants')

}
```