INSTITUTO POLITÉCNICO DO CÁVADO E DO AVE

Engenharia em Desenvolvimento de Jogos Digitais

Relatório

Computação Móvel

Paulo André Moreira Macedo 17011

Luis Manuel Gonçalves da Silva 17012

Daniel Filipe Marques Fernandes 17014

Barcelos, janeiro 2020

Cofinanciado por:

POCH

PORTUGAL 2020

UNIÃO EUROPEIA

Fundo Social Europeu

Instituto Politécnico do Cávado e do Ave

Engenharia em Desenvolvimento de Jogos Digitais

Relatório como requisito do projeto da Cadeira Computação Móvel do Curso de Engenharia em Desenvolvimento de Jogos Digitais, sob a orientação do docente Lourenço Gomes.

Barcelos, janeiro 2020

Cofinanciado por:

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu

Índice

Índice de Ilustrações

Introdução

Este relatório constitui a fundamentação teórica sofre o nosso projeto e pretende dar a conhecer o mesmo por nós desenvolvido.

O projeto desenvolvido é então uma aplicação direcionada a uma florista na qual o utilizador pode criar, alterar e/ou remover Bouquets personalizados com o número de flores ao seu agrado. Permite ainda a seleção dos Bouquets a querer adquirir, dirigindo-se a um menu de Checkout onde no qual pode ainda decidir a quantidade desejada de cada Bouquet e ainda o seu pagamento direto por Paypal ou algum cartão a essa conta associado.

Desenvolvimento

Capítulo I – Modelo de Dados

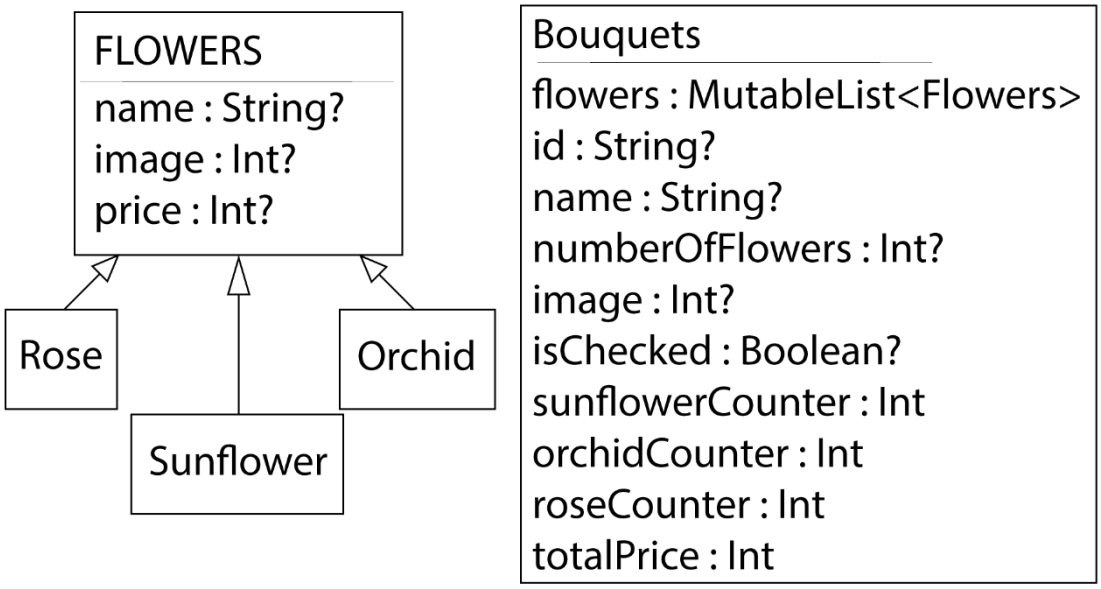


Figura 1 - Modelo de Dados

Os modelos de dados acima representados constituem a base na qual a aplicação se baseia.

A acrescentar, ambas as Classes herdam Serializable visto que é necessário o passar destes objetos entre Activities.

Capítulo II – Esquema Geral (Lógica)

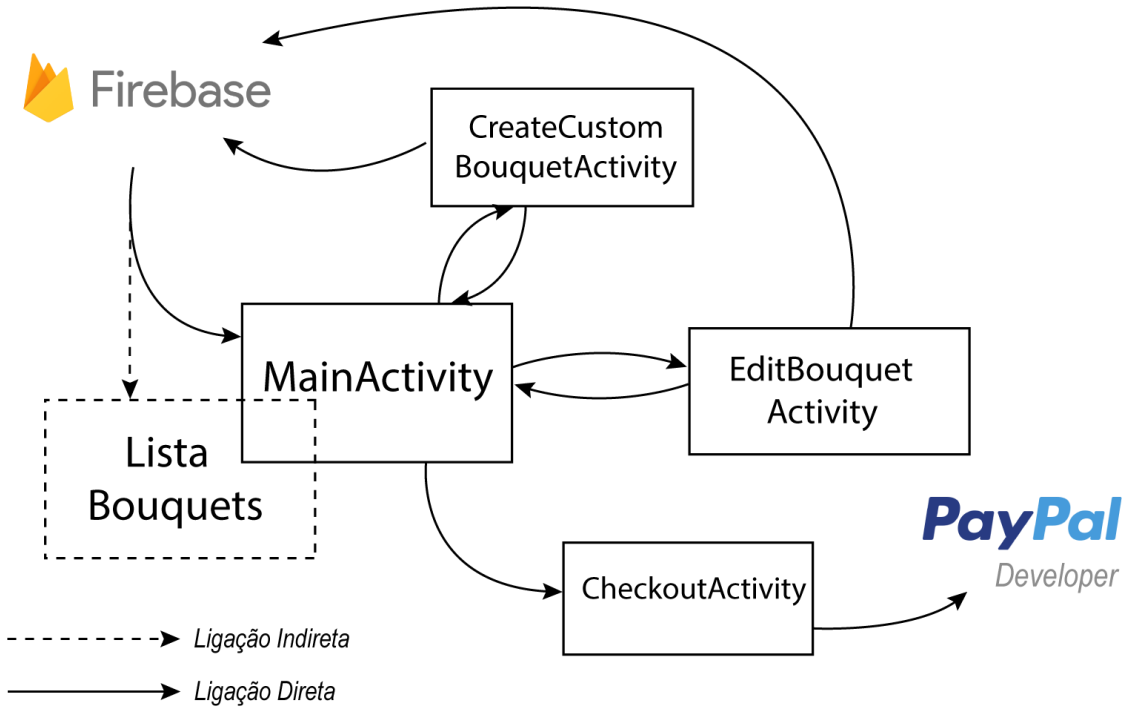


Figura 2 - Esquema Geral

A MainActivity é responsável pela gestão dos dados recebidos pela FireBase e pelas outras Activities para a alteração da Lista nela contida. É por fim esta que gere quais os elementos a serem enviados para a CheckoutActivity que por sua vez tem acesso à API do Paypal para o pagamento final.

Capítulo III – Main Activity

Método predefinedBouquetsCreation

```
// Creates 3 predefined Bouquets that are not stored in Firebase
private fun predefinedBouquetsCreation() {
    var flowersListForPredefinedBouquet: MutableList<Flowers> = ArrayList<Flowers>()

    //First Bouquet- 100 sunflowers
    for(x in 0..99  )
    {
        flowersListForPredefinedBouquet.add(Sunflower())
    }
    bouquetList.add(Bouquets( name: "Shooting Star",flowersListForPredefinedBouquet,R.drawable.shootingstar))

    flowersListForPredefinedBouquet.clear()

    //SecondBouquet- 100 Roses
    for(x in 0..99  )
    {
        flowersListForPredefinedBouquet.add(Rose())
    }
    bouquetList.add(Bouquets( name: "Bloody Mary", flowersListForPredefinedBouquet, R.drawable.bloodymary))

    flowersListForPredefinedBouquet.clear()


    //Third Bouquet- 100 Orchids
    for(x in 0..99  )
    {
        flowersListForPredefinedBouquet.add(Orchid())
    }
    bouquetList.add(Bouquets( name: "Venus", flowersListForPredefinedBouquet, R.drawable.venus))

    flowersListForPredefinedBouquet.clear()
}
```

Figura 3 - Metodo predefinedBouquetsCreation

Método responsável pela criação de três Bouquets predefinidos não guardados na FireBase.

Método onCreate

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)
    getSupportActionBar()!!.setTitle("D.Lurdes");

    // Gets reference from correspondent node in Firebase of Bouquet storage
    ref = FirebaseDatabase.getInstance().getReference( path: "Bouquets")

    // Created predefined bouquets(not stored in Firebase)
    predefinedBouquetsCreation()

    // Sets up adapter for the list
    bouquetListView.adapter = BouquetAdapter()

    // Reads custom bouquets from Firebase
    readingFirebaseData()

    // Calls and manages result from CreateCustomBouquetActivity
    addNewBouquetManager()

    // Manages the button for the CheckoutActivity
    checkoutManager()
}
```

Figura 4 - Método onCreate

Método responsável pelo chamar dos outros métodos essenciais para a Activity e pelo inicializar e definição a valores iniciais da mesma.

Método readingFirebaseData

```
// Reads the data from the associated Firebase and stores them in the list
private fun readingFirebaseData(){

    ref.addValueEventListener(object : ValueEventListener{
        override fun onCancelled(p0: DatabaseError) {
            TODO( reason: "not implemented") //To change body of created functions use File | Settings | File Templates.
        }
        override fun onDataChange(p0: DataSnapshot) {
            if(p0.exists()){

                for (h in p0.children){

                    // Bool to check if the node is'nt already stored in the list
                    var alreadyInList : Boolean = false

                    // Gets current node Bouquet
                    var bouquetInCurrentNode = h.getValue(Bouquets::class.java)


                    // Checks the list for a bouquet with the same id
                    for(b in bouquetList){

                        // If it finds one it changes the bool variable to true
                        if(b.id != null && b.id == bouquetInCurrentNode!!.id ) {

                            alreadyInList = true
                            break
                        }
                    }

                    // If the bouquet in the current node of the firebase is'nt stored in the list it stores it
                    if(alreadyInList == false){

                        bouquetList.add(bouquetInCurrentNode!!)
                    }
                }
                // Updates listView
                bouquetListView.adapter = BouquetAdapter()
            }
        }
    })
}
```

Figura 5 - Método readingFirebaseData

Método responsável pela leitura de dados da FireBase para armazenamento da lista, tendo sido utilizada uma condição que compare o id de cada node na FireBase à lista para a verificação se esta já se encontra guardado ou não.

Metodo addNewBouquetManager

```
// Manages the button for the CreateCustomBouquetActivity
 private fun addNewBouquetManager(){

     addNewBouquet.setOnClickListener { it: View!

         val intent = Intent( packageContext: this@MainActivity, CreateCustomBouquetActivity::class.java)

         startActivity(intent)
     }
 }
```

Figura 6 - Método addNewBouquetsManager

Método responsável pela gestão do botão responsável da inicialização da CreateNewBouquetActivity.

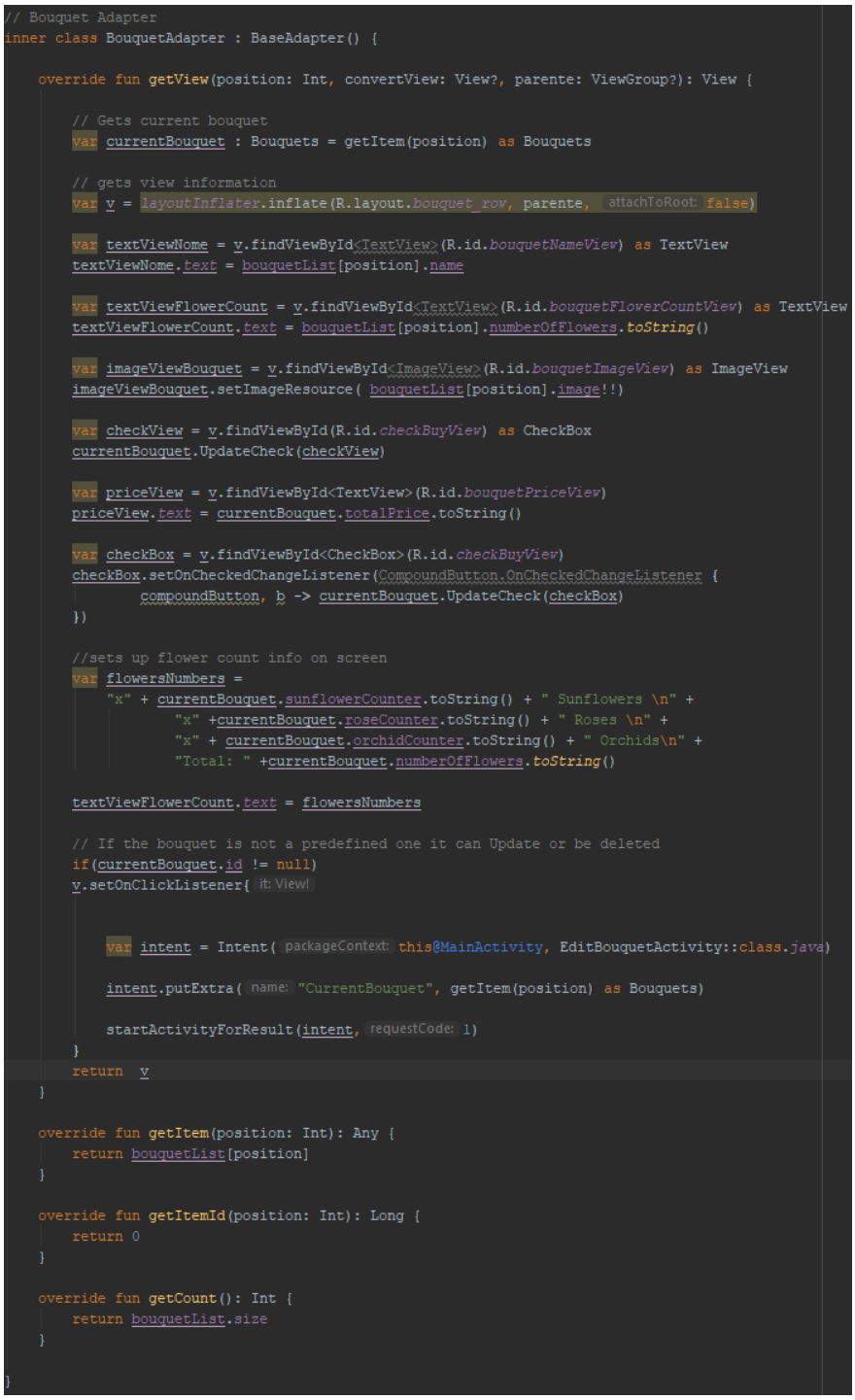
Metodo checkoutManager

```
// Manages the button for the CheckoutActivity
private fun checkoutManager(){

    checkoutButtonView.setOnClickListener{ it: View!
        val intent = Intent( packageContext: this@MainActivity, CheckoutActivity::class.java)

        var bouquetCounter : Int = 0

        for(checkBouquet in bouquetList){

            if(checkBouquet.isChecked == true){

                bouquetCounter++

                var bouquetKey = "BouquetNumber" + bouquetCounter

                intent.putExtra(bouquetKey, checkBouquet)
            }
        }

        intent.putExtra( name: "BouquetCounter", bouquetCounter)
        startActivity(intent)
    }
}
```

Figura 7 - Método checkoutManager

Método responsável pela gestão do botão responsável pela inicialização da checkoutActivity, mandando para esta todos os elementos selecionados.

Bouquet Adapter



```
// Bouquet Adapter
inner class BouquetAdapter : BaseAdapter() {

    override fun getView(position: Int, convertView: View?, parente: ViewGroup?): View {

        // Gets current bouquet
        var currentBouquet : Bouquets = getItem(position) as Bouquets

        // gets view information
        var v = layoutInflater.inflate(R.layout.bouquet_row, parente,  attachToRoot: false)

        var textViewNome = v.findViewById<TextView>(R.id.bouquetNameView) as TextView
        textViewNome.text = bouquetList[position].name

        var textViewFlowerCount = v.findViewById<TextView>(R.id.bouquetFlowerCountView) as TextView
        textViewFlowerCount.text = bouquetList[position].numberOfFlowers.toString()

        var imageViewBouquet = v.findViewById<ImageView>(R.id.bouquetImageView) as ImageView
        imageViewBouquet.setImageResource( bouquetList[position].image!!)

        var checkView = v.findViewById(R.id.checkBuyView) as CheckBox
        currentBouquet.UpdateCheck(checkView)

        var priceView = v.findViewById<TextView>(R.id.bouquetPriceView)
        priceView.text = currentBouquet.totalPrice.toString()

        var checkBox = v.findViewById<CheckBox>(R.id.checkBuyView)
        checkBox.setOnCheckedChangeListener(CompoundButton.OnCheckedChangeListener {
                compoundButton, b -> currentBouquet.UpdateCheck(checkBox)
        })

        //sets up flower count info on screen
        var flowersNumbers =
            "x" + currentBouquet.sunflowerCounter.toString() + " Sunflowers \n" +
                    "x" +currentBouquet.roseCounter.toString() + " Roses \n" +
                    "x" + currentBouquet.orchidCounter.toString() + " Orchids\n" +
                    "Total: " +currentBouquet.numberOfFlowers.toString()

        textViewFlowerCount.text = flowersNumbers

        // If the bouquet is not a predefined one it can Update or be deleted
        if(currentBouquet.id != null)
        v.setOnClickListener{ it: View!


            var intent = Intent( packageContext: this@MainActivity, EditBouquetActivity::class.java)

            intent.putExtra( name: "CurrentBouquet", getItem(position) as Bouquets)

            startActivityForResult(intent,  requestCode: 1)
        }
        return  v
    }

    override fun getItem(position: Int): Any {
        return bouquetList[position]
    }

    override fun getItemId(position: Int): Long {
        return 0
    }

    override fun getCount(): Int {
        return bouquetList.size
    }

}
```

Figura 8 - Bouquet Adapter

Adapter customizado para a apresentação da lista de Bouquets e responsável pela gestão do Click em elementos da lista (exceto os predefinidos). É este que inicia a EditBouquetActivity para a obtenção de resultados.

Metodo onActivityResult

```
// Manages activity results from EditBouquetActivity
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    // Result from EditBouquetActivity
     if(requestCode == 1 && resultCode == Activity.RESULT_OK){

        // Gets which action was made (Update)
        var typeOfReturn : String = data?.getStringExtra( name: "TypeOfReturn")!!

        // UPDATE ACTION
         if(typeOfReturn == "UPDATE"){

            // Gets data returned from the EditBouquetActivity

            // Gets the id of the bouquet to update
            var bouquetIdToUpdate = data?.getStringExtra( name: "BouquetToUpdateId")

            // Gets the updated bouquet
            var bouquetUpdated =  data?.getSerializableExtra( name: "BouquetForUpdate") as Bouquets

            // Variable responsible for storing the index of the bouquet in the list to update
            var indexToSubstitute : Int = 0

            for(b in bouquetList)
            {
                if(b.id == bouquetIdToUpdate) indexToSubstitute = bouquetList.indexOf(b)
            }

            // Substitutes the bouquet with the same id
            bouquetList[indexToSubstitute] = bouquetUpdated

            // Small message pop up to show it went sucessfully
            Toast.makeText( context: this, text: "Bouquet Updated", Toast.LENGTH_LONG).show()
        }

        // DELETE ACTION
        else if(typeOfReturn == "DELETE"){

             // Gets the data of the id of the bouquet to remove
            var bouquetIdToRemove = data?.getStringExtra( name: "BouquetToRemoveId")

             // Searches the bouquet list for the bouquet with the same id to be removed
            for(b in bouquetList){

                if(b.id == bouquetIdToRemove){

                    bouquetList.remove(b)
                    break
                }
            }

             // Makes small message pop up
             Toast.makeText( context: this, text: "Bouquet Deleted", Toast.LENGTH_LONG).show()
        }

        // Updates listView
        bouquetListView.adapter = BouquetAdapter()

    }
}
```

Figura 9 - Método onActivityResult

Método responsável obtenção e manuseamento de dados dependendo dos retornados pela EditBouquetActivity.

Capítulo IV – FlowerSelection

Devido à linguagem Kotlin não fornecer a alteração por Reference de variáveis de PrimitiveTypes foi necessário a criação de uma Classe que servisse para o armazenamento dos valores selecionados pelo utilizador no menu e para a criação da lista de Flores predefinidas (Disponíveis na loja).

```
// List for show which flowers are available
var allDifferentFlowerTypes: MutableList<Flowers> = ArrayList<Flowers>()

// Counters fo each type of flower selected
 var numberSunflowerSelected = 0
 var numberRoseSelected = 0
 var numberOrchidSelected = 0

constructor(){

    PredefinedListCreation()
}

constructor(bouquetReceivedForEdit : Bouquets){

    PredefinedListCreation()
    CountersDefinedByPreviousCreatedBouquet(bouquetReceivedForEdit)
}

private fun CountersDefinedByPreviousCreatedBouquet(bouquetReceivedForEdit : Bouquets){

    numberSunflowerSelected = bouquetReceivedForEdit.sunflowerCounter
    numberRoseSelected = bouquetReceivedForEdit.roseCounter
    numberOrchidSelected = bouquetReceivedForEdit.orchidCounter
}

// Creates List with all different type of flowers
private fun PredefinedListCreation(){

    allDifferentFlowerTypes.add(Sunflower())
    allDifferentFlowerTypes.add((Rose()))
    allDifferentFlowerTypes.add((Orchid()))
}
```

Figura 10 - Classe FlowersSelection

Esta Classe serve-se de dois Construtores devido à necessidade dos valores na EditBouquetActivity terem de ser inicializados com valores recebidos (variáveis), enquanto que no CreateCustomBouquetActivity estes valores iniciam a zero.

Capítulo V – CreateCustomBoquetActivity

Método onCreate

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_create_custom_bouquet)
    getSupportActionBar()!!.setTitle("D.Lurdes");

    // Gets reference from correspondent node in firebase of Bouquet storage
    ref = FirebaseDatabase.getInstance().getReference( path: "Bouquets")

    // Sets up custom adapter
    allFlowerTypeView.adapter = FlowerTypeListAdapter()

    // Manages confirmButton action
    confirmButtonManagement()
}
```

Figura 11 - Método onCreate

Método responsável pelo chamar dos outros métodos essenciais para a Activity e pelo inicializar e definição a valores iniciais da mesma.

Método confirmButtonManagement

```
// Manages Confirm Button action
private fun confirmButtonManagement(){

    // Manages button click
    confirmAdd.setOnClickListener() { it: View!

        // Creates Bouquet from selected flowers
        var customBouquet = createCustomBouquet()

        // If there were flowers selected
       if(customBouquet.numberOfFlowers!! > 0){

           // Gets new id in the Firebase for the new created bouquet
           val bouquetId = ref.push().key
           customBouquet.id = bouquetId

           // Adds the new bouquet to the Firebase
           ref.child(bouquetId!!).setValue(customBouquet).addOnCompleteListener{ it: Task<Void!>

               // Makes pop up message confirming the save
               Toast.makeText( context: this, text: "Bouquet Saved!", Toast.LENGTH_LONG).show()
           }
       }
        else{

           // Makes pop up message telling there weren't flowers selected so the bouquet was not saved
           Toast.makeText( context: this, text: "No Flowers Selected\nBouquet Not Saved!", Toast.LENGTH_LONG).show()
       }
        // Closes current activity and return to main activity
        finish()
    }
}
```

Figura 12 - Método confirmButtonManagement

Método responsável pelo guardar o Bouquet recém-criado com as flores selecionadas pelo utilizar na Firebase com um ID único gerado pela mesma. Este, porém, só é guardado se existirem flores selecionadas.

Metodo createCustomBouquet

```
// Creates custom Bouquet from selected flowers
private fun createCustomBouquet() : Bouquets{

    // Creates temporary flower list for custom bouquet creation
    var flowerListForCustomBouquet : MutableList<Flowers> = ArrayList<Flowers>()

    for(x in 1..flowerSelectionManager.numberSunflowerSelected) flowerListForCustomBouquet.add(Sunflower())
    for(x in 1..flowerSelectionManager.numberRoseSelected) flowerListForCustomBouquet.add(Rose())
    for(x in 1..flowerSelectionManager.numberOrchidSelected) flowerListForCustomBouquet.add(Orchid())

    return  Bouquets( name: "Custom Bouquet", flowerListForCustomBouquet, imageChoosing())
}
```

Figura 13 - Método createCustomBouquet

Método responsável pela criação do Bouquet com as flores selecionadas pelo utilizador.

Método imageChoosing

```
private fun imageChoosing() : Int{

     var selectedImageforShow : Int

    // Priority list in case its equal number-> Venus - BloodyMary - Shooting Star

    if (flowerSelectionManager.numberOrchidSelected >= flowerSelectionManager.numberRoseSelected)
    {
        if (flowerSelectionManager.numberOrchidSelected >= flowerSelectionManager.numberSunflowerSelected)
        {
            selectedImageforShow = R.drawable.venus
        }
        else selectedImageforShow = R.drawable.shootingstar
    }
    else
    {
        if (flowerSelectionManager.numberRoseSelected >= flowerSelectionManager.numberSunflowerSelected)
        {
            selectedImageforShow = R.drawable.bloodymary
        }
        else selectedImageforShow = R.drawable.shootingstar
    }
    return selectedImageforShow
}
```

Figura 14 - Método imageChoosing

Método responsável pela gestão da imagem para o CustomBouquet dependendo da flor mais selecionada.

FlowerTypeListAdapter

```
inner class FlowerTypeListAdapter : BaseAdapter() {

    override fun getView(position: Int, convertView: View?, parente: ViewGroup?): View {

        var currentFlower : Flowers = getItem(position) as Flowers

        // gets view information
        var v = layoutInflater.inflate(R.layout.flowertype_row, parente,  attachToRoot: false)

        var textViewNome = v.findViewById<TextView>(R.id.flowerTypeNameView)
        textViewNome.text = currentFlower.name.toString()


        var flowerImageView = v.findViewById<ImageView>(R.id.flowerTypeImageView)
        flowerImageView.setImageResource( currentFlower.image!!)



        // Gets adding and removing flowers buttons
        var minusButtonView = v.findViewById<Button>(R.id.minusButton) as Button
        var plusButtonView = v.findViewById<Button>(R.id.plusButton) as Button


        // Gets current flower type number
        var currentFlowerTypeSelectionView =  v.findViewById(R.id.flowerTypeNumberSelection) as EditText


        var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()

        currentFlowerTypeNumberStoring(currentNumber,currentFlower)

        minusButtonView.setOnClickListener { it: View!

            var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()
            currentNumber--


            currentFlowerTypeSelectionView.text = Editable.Factory.getInstance().newEditable(currentNumber.toString())

            currentFlowerTypeNumberStoring(currentNumber,currentFlower)
        }

        plusButtonView.setOnClickListener { it: View!

            var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()

            currentNumber++

            currentFlowerTypeSelectionView.text =  Editable.Factory.getInstance().newEditable(currentNumber.toString())

            currentFlowerTypeNumberStoring(currentNumber,currentFlower)
        }
        return  v
    }

    private fun currentFlowerTypeNumberStoring (currentNumber : Int, currentFlower : Flowers) {

        when(currentFlower){

            is Sunflower -> {

                flowerSelectionManager.numberSunflowerSelected = currentNumber
            }
            is Rose ->{

                flowerSelectionManager.numberRoseSelected = currentNumber
            }
            is Orchid ->{

                flowerSelectionManager.numberOrchidSelected = currentNumber
            }
        }
    }
```

Figura 15 - FlowerTypeListAdapter

Adapter responsável pela apresentação da lista de flores com a sua respetiva seleção e alteração dos valores conforme o utilizador os altera.

Capítulo VI – Classe EditBouquetActivity

Método onCreate

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.edit_activity)

    // Gets bouquet selected
    var bouquetReceived = intent.getSerializableExtra( name: "CurrentBouquet") as Bouquets

    // Gets reference from correspondent node in firebase of Bouquet storage
     ref = FirebaseDatabase.getInstance().getReference( path: "Bouquets")

    // Creates a flower selection manager with the starting values as the ones of the bouquet
    flowerSelectionManager = FlowerSelection(bouquetReceived)

    // Gets the toolbar tiltle to be the same as the selected bouquet
    getSupportActionBar()!!.setTitle(bouquetReceived.name)

    // Updates listView
    allFlowerTypeForEditView.adapter = FlowerTypeListUpdateAdapter()

    // Manages the updateButton click and substitutes value in Firebase
    updateButton.setOnClickListener{ it: View!

        // Gets updated bouquet object and associates with the id of the one to be updated
        var bouquetUpdated = updateBouquet()
        bouquetUpdated.id = bouquetReceived.id

        // Substitutes the bouquet in the Firebase
        ref.child(bouquetReceived.id!!).setValue({})


        // Creates intent to return necessary info
        var resultIntent = Intent()

        // Returns the info to know which action the user chose
        resultIntent.putExtra( name: "TypeOfReturn",  value: "UPDATE")

        // Returns the necessary info to update the bouquet in the MainActivity list of bouquets
        resultIntent.putExtra( name: "BouquetToUpdateId",bouquetUpdated.id )
        resultIntent.putExtra( name: "BouquetForUpdate", bouquetUpdated)

        setResult(Activity.RESULT_OK, resultIntent)

        finish()
    }

    deleteButton.setOnClickListener{ it: View!

        // Removes the node from the Firebase of the selected bouquet
        ref.child(bouquetReceived.id!!).removeValue()

        // Intent made to return the id of the node removed so it can be removed from the list aswell
        var resultIntent = Intent()

        resultIntent.putExtra( name: "TypeOfReturn",  value: "DELETE")
        resultIntent.putExtra( name: "BouquetToRemoveId",bouquetReceived.id )

        setResult(Activity.RESULT_OK, resultIntent)

        finish()
    }
}
```

Figura 16 - Método onCreate

Método responsável pelo chamar dos outros métodos essenciais para a Activity e pelo inicializar e definição a valores iniciais da mesma. Além disso é também responsável pela gestão da ação que o utilizador pretende executar (Update ou deleção do Bouquet selecionado).

Método updateBouquet & imageChoosing

```
private fun updateBouquet() : Bouquets{

    //creates temporary flower list for custom bouquet creation
    var flowerListForCustomBouquet : MutableList<Flowers> = ArrayList<Flowers>()

    for(x in 1..flowerSelectionManager.numberSunflowerSelected) flowerListForCustomBouquet.add(Sunflower())
    for(x in 1..flowerSelectionManager.numberRoseSelected) flowerListForCustomBouquet.add(Rose())
    for(x in 1..flowerSelectionManager.numberOrchidSelected) flowerListForCustomBouquet.add(Orchid())

    return Bouquets( name: "Custom Bouquet", flowerListForCustomBouquet, imageChoosing())
}

private fun imageChoosing() : Int{

    var selectedImageforShow : Int

    // Priority list in case its equal number-> Venus - BloodyMary - Shooting Star

    if (flowerSelectionManager.numberOrchidSelected >= flowerSelectionManager.numberRoseSelected)
    {
        if (flowerSelectionManager.numberOrchidSelected >= flowerSelectionManager.numberSunflowerSelected)
        {
            selectedImageforShow = R.drawable.venus
        }
        else selectedImageforShow = R.drawable.shootingstar
    }
    else
    {
        if (flowerSelectionManager.numberRoseSelected >= flowerSelectionManager.numberSunflowerSelected)
        {
            selectedImageforShow = R.drawable.bloodymary
        }
        else selectedImageforShow = R.drawable.shootingstar
    }
    return selectedImageforShow
}
```

Figura 17 - Método updateBouquet & imageChoosing

Métodos de função semelhantes aos da página 17.

FlowerTypeListUpdateAdapter

```
inner class FlowerTypeListUpdateAdapter : BaseAdapter() {

    override fun getView(position: Int, convertView: View?, parente: ViewGroup?): View {

        var currentFlower : Flowers = getItem(position) as Flowers

        // gets view information
        var v = layoutInflater.inflate(R.layout.flowertype_row, parente,  attachToRoot: false)

        var textViewNome = v.findViewById<TextView>(R.id.flowerTypeNameView)
        textViewNome.text = currentFlower.name.toString()

        var flowerImageView = v.findViewById<ImageView>(R.id.flowerTypeImageView)
        flowerImageView.setImageResource( currentFlower.image!!)

        // Gets adding and removing flowers buttons
        var minusButtonView = v.findViewById<Button>(R.id.minusButton) as Button
        var plusButtonView = v.findViewById<Button>(R.id.plusButton) as Button

        // Gets current flower type number
        var currentFlowerTypeSelectionView =  v.findViewById(R.id.flowerTypeNumberSelection) as EditText

        when(currentFlower){

            is Sunflower -> currentFlowerTypeSelectionView.text = Editable.Factory.getInstance().newEditable(flowerSelectionManager.numberSunflowerSelected.toString())
            is Rose -> currentFlowerTypeSelectionView.text = Editable.Factory.getInstance().newEditable(flowerSelectionManager.numberRoseSelected.toString())
            is Orchid -> currentFlowerTypeSelectionView.text = Editable.Factory.getInstance().newEditable(flowerSelectionManager.numberOrchidSelected.toString())

        }

        var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()

        currentFlowerTypeNumberStoring(currentNumber,currentFlower)

        minusButtonView.setOnClickListener { it: View!

            var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()
            currentNumber--


            currentFlowerTypeSelectionView.text = Editable.Factory.getInstance().newEditable(currentNumber.toString())

            currentFlowerTypeNumberStoring(currentNumber,currentFlower)
        }

        plusButtonView.setOnClickListener { it: View!

            var currentNumber = currentFlowerTypeSelectionView.text.toString().toInt()

            currentNumber++


            currentFlowerTypeSelectionView.text =  Editable.Factory.getInstance().newEditable(currentNumber.toString())

            currentFlowerTypeNumberStoring(currentNumber,currentFlower)

        }
        return  v
    }

    private fun currentFlowerTypeNumberStoring (currentNumber : Int, currentFlower : Flowers) {

        when(currentFlower){

            is Sunflower -> {

                flowerSelectionManager.numberSunflowerSelected = currentNumber

            }
            is Rose ->{

                flowerSelectionManager.numberRoseSelected = currentNumber
            }
            is Orchid ->{

                flowerSelectionManager.numberOrchidSelected = currentNumber
            }
        }
    }

    override fun getItem(position: Int): Any {
        return flowerSelectionManager.allDifferentFlowerTypes[position]
    }
```

Figura 18 – Método FlowerTypeListUpdateAdapter

Adapter responsável pela apresentação da lista de flores com a sua respetiva seleção e alteração dos valores conforme o utilizador os altera (Neste caso os valores iniciam com os valores do Bouquet selecionado).

Capítulo VII – UserInfo

```
class UserInfo
{
    //Paypal acc:
    //dlurdes@inwmail.net
    //Dlurdesflorista!1

    //bus-dlurdes@hotmail.com (business)
    //p-dlurdes@hotmail.com (client used for testing)
    //p2-dlurdes@hotmail.com
    //123456789

    //developer.paypal.com


    // Account where the money is send to
    companion object {
        var client_id :String = "Ab_pVm2LgBAGcKo4NILBTQ_cwroUGQi9J0o4evZ1XzfhcvinEraft6dhrQdFXtNaGQ6VeKGrK_IIEt9F"
    }

}
```

Figura 19 - Classe UserInfo

Classe responsável pelo armazenamento da conta, a qual recebe pagamentos.

Capítulo VIII – CheckoutActivity

Metodo onCreate

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_checkout)

    // Gets the number of selected bouquets
    var checkedBouquetCounter = intent.getIntExtra( name: "BouquetCounter", defaultValue: 0)

    for(x in 1.. checkedBouquetCounter){


        // Gets all bouquets sent through intent (the ones selected) and adds them to the list
        var bouquetKey = "BouquetNumber" + x
        var currentBouquet = intent.getSerializableExtra(bouquetKey) as Bouquets

        checkoutBouquetList.add(currentBouquet)
    }

    // Initial total price(1 of each bouquet)
    for(b in checkoutBouquetList){

        priceToPay += b.totalPrice

    }
    paypalButton.text = "PayPal: " + priceToPay + "€"

    config = PayPalConfiguration().environment(PayPalConfiguration.ENVIRONMENT_SANDBOX).clientId(UserInfo.client_id)
    var intent = Intent( packageContext: this, PayPalService::class.java)
    intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION,config)
    startService(intent)

    paypalButton.setOnClickListener{ it: View!

        amount = priceToPay.toDouble()

        var payment = PayPalPayment(BigDecimal.valueOf(amount),"EUR","DLurdes", PayPalPayment.PAYMENT_INTENT_SALE)
        var intent = Intent( packageContext: this, PaymentActivity::class.java)
        intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION, config)
        intent.putExtra(PaymentActivity.EXTRA_PAYMENT, payment)
        startActivityForResult(intent,  requestCode: 1)
    }

    // Sets up custom adapter
    checkoutListView.adapter = CheckoutListAdapter()
}
```

Figura 20 - Método onCreate

Método responsável pelo chamar dos outros métodos essenciais para a Activity e pelo inicializar e definição a valores iniciais da mesma. Além disso é o responsável pela busca e armazenamento de dados recebidos pelo Intent da MainActivity e pela inicialização e comunicação com a API do Paypal.

Método totalPriceUpdate

```
private fun totalPriceUpdate(valueToRemove : Int, valueToAdd: Int){

    priceToPay -= valueToRemove
    priceToPay += valueToAdd
    paypalButton.text = "PayPal: " + priceToPay + "€"

}
```

Figura 21 - Método totalPriceUpdate

Método responsável por dar update ao preço total a pagar quando são feitas alterações no número de Bouquets selecionados.

Metodo onActivityResult

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == 1){

        if (resultCode == Activity.RESULT_OK)
        {
            Toast.makeText( context: this, text: "Transaction Completed!", Toast.LENGTH_LONG).show()
            finish()
        }
        else
        {
            Toast.makeText( context: this, text: "Transaction Failed!", Toast.LENGTH_LONG).show()
            finish()
        }
    }
}
```

Figura 22 - Método onActivityResult

Método responsável por comunicar ao utilizador se a transação foi bem ou malsucedida.

CheckoutListAdapter

```
inner class CheckoutListAdapter : BaseAdapter() {

    override fun getView(position: Int, convertView: View?, parente: ViewGroup?): View {

        var currentBouquet : Bouquets = getItem(position) as Bouquets

        // Gets View information
        var v = layoutInflater.inflate(R.layout.checkout_row, parente,  attachToRoot: false)

        var textViewNome = v.findViewById<TextView>(R.id.checkoutBouquetNameView)
        textViewNome.text = currentBouquet.name.toString()

        var bouquetImageView = v.findViewById<ImageView>(R.id.checkoutBouquetImageView)
        bouquetImageView.setImageResource( currentBouquet.image!!)

        // Gets adding and removing flowers buttons
        var minusButtonView = v.findViewById<Button>(R.id.checkoutMinusButton) as Button
        var plusButtonView = v.findViewById<Button>(R.id.checkoutPlusButton) as Button

        // Gets current flower type number

        var currentBouquetQuantityView =  v.findViewById(R.id.checkoutBouquetQuantity) as TextView

        var currentNumber = currentBouquetQuantityView.text.toString().toInt()

        var bouquetPriceView = v.findViewById<TextView>(R.id.checkoutBouquetPrice)
        var totalPriceOfCurrentBouquetQuantity = currentNumber *  currentBouquet.totalPrice
        bouquetPriceView.text = totalPriceOfCurrentBouquetQuantity.toString()

        //Sets up flower count info on screen
        var checkoutBouquetFlowerDescriptionView = v.findViewById<TextView>(R.id.checkoutBouquetFlowerDescription)
        var flowersNumbers =
            "x" + currentBouquet.sunflowerCounter.toString() + " Sunflowers \n" +
                    "x" +currentBouquet.roseCounter.toString() + " Roses \n" +
                    "x" + currentBouquet.orchidCounter.toString() + " Orchids\n" +
                    "Total: " +currentBouquet.numberOfFlowers.toString()

        checkoutBouquetFlowerDescriptionView.text = flowersNumbers

        minusButtonView.setOnClickListener { it: View!

            var valueToRemove = currentNumber * currentBouquet.totalPrice

            currentNumber--

            var totalPriceOfCurrentBouquetQuantity = currentNumber *  currentBouquet.totalPrice
            bouquetPriceView.text = totalPriceOfCurrentBouquetQuantity.toString()

            totalPriceUpdate(valueToRemove,totalPriceOfCurrentBouquetQuantity)

            currentBouquetQuantityView.text = currentNumber.toString()
        }

        plusButtonView.setOnClickListener { it: View!

            var valueToRemove = currentNumber * currentBouquet.totalPrice

            currentNumber++

            var totalPriceOfCurrentBouquetQuantity = currentNumber *  currentBouquet.totalPrice
            bouquetPriceView.text = totalPriceOfCurrentBouquetQuantity.toString()

            totalPriceUpdate(valueToRemove,totalPriceOfCurrentBouquetQuantity)

            currentBouquetQuantityView.text =  currentNumber.toString()
        }
        return  v
    }
```

Figura 23 - Adapter CheckoutListAdapter

Adapter responsável pela apresentação da lista de Bouquets a comprar com a sua respetiva seleção e alteração dos valores conforme o utilizador os altera, com os respetivos preços e o total a pagar pelo todo.

Capítulo IX – Analise Critica

Esta análise crítica pretende fazer uma reflexão sobre o projeto. Tendo em conta os nossos objetivos no início do desenvolvimento do projeto, cumprimos e superamos todas as nossas metas.

Conclusão

O nosso relatório consistiu na realização do projeto, para conseguir concluir este projeto foi preciso ter uma grande capacidade de autonomia e persistência, aspetos esses que se foram desenvolvendo à medida que o concebíamos, planeávamos e executávamos.

A execução do projeto envolveu um grande esforço e dedicação. Para além das competências técnicas e diversas capacidades que o trabalho exigiu, pensamos que foi benéfico para nós a nível profissional e a nível social, uma vez que no futuro iremos enfrentar outros projetos tão ou mais importante que este.

Naturalmente que, no decorrer da realização do projeto sentimos diversas dificuldades, as quais só puderam ser ultrapassadas com grande esforço da nossa parte.

Web Grafia

General:

https://www.youtube.com/watch?v=F9UC9DY-vIU

Layout:

https://www.youtube.com/watch?v=AKvYOpbDYKA

Lists and Arrays:

https://www.youtube.com/watch?v=Je_YXshSFmY

https://www.youtube.com/watch?v=_SgmgA7Kz2g

ListView:

https://www.youtube.com/watch?v=EwwdQt3_fFU

https://www.youtube.com/watch?v=95QWxTZG_Z0

https://www.youtube.com/watch?v=BkmXPDY-1GE

https://www.youtube.com/watch?v=fwwu2mDD4cw

Intent:

https://www.youtube.com/watch?v=S1isQRnYAF4

https://www.youtube.com/watch?v=JjvY6bExiyg

https://www.youtube.com/watch?v=dcd6stWGObk

https://www.youtube.com/watch?v=JDxuBbsua_E

https://www.youtube.com/watch?v=2W41M9fWf6I

Object Keyword ("Static"):

https://www.youtube.com/watch?v=kH4pSAFSheU

https://www.youtube.com/watch?v=mK-0Zdjhcuk

Firebase API:

https://www.youtube.com/watch?v=I485b7LzYkM

https://www.youtube.com/watch?v=ZB1liwuQCP8

https://www.youtube.com/watch?v=kc3LVeCDy14

Paypal API:

https://www.youtube.com/watch?v=YO85GJoxkqE

https://www.youtube.com/watch?v=fYvXblhsjIg&t

https://www.youtube.com/watch?v=jrhPAiwDv1U