Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по проекту «Криптобиржа»

«Инструменты разработки мобильных приложений»

Выполнил:
студент группы 381906-1
Пудовкин А. В.

Нижний Новгород
2022

Оглавление

Цель задачи

Создать приложение-игру «Криптобиржа» с рандомным поведением цен криптокоинов и подсчетом заработка.

Описание программной реализации

1. Стартовый капитал

```
property int startingAmount: 10000
```

2. Массив с информацией о коинах

```
property var coins: [
    {
        "name": "BTC",
        "lastPrice": 20690.88,
        "price": 20690.88,
        "color": "orange",
    },
    {
        "name": "ETH",
        "lastPrice": 1517.35,
        "price": 1517.35,
        "color": "darkslategray",
    },
    {
        "name": "LTC",
        "lastPrice": 83.56,
        "price": 83.56,
        "color": "indigo",
    },
    {
        "name": "BNB",
        "lastPrice": 286.59,
        "price": 286.59,
        "color": "gold",
    },
    {
        "name": "SOL",
        "lastPrice": 20.78,
        "price": 20.78,
        "color": "deepskyblue",
    },
]
```

3. Массив-состяние кошелька игрока

```
property var wallet: [
{
    "name": "$USD",
    "sum": startingAmount,
},
{
    "name": "BTC",
    "sum": 0,
},
```

```
    {
        "name": "ETH",
        "sum": 0,
    },
    {
        "name": "LTC",
        "sum": 0,
    },
    {
        "name": "BNB",
        "sum": 0,
    },
    {
        "name": "SOL",
        "sum": 0,
    },
]
```

4. **Grid** и элементов **Repeater**, который отрисовывает все элементы управления коинами.

```
Grid {
    anchors.horizontalCenter: parent.horizontalCenter
    columns: 2
    spacing: 20
    Repeater {
        id: repeater
        model: coins
        delegate: Rectangle {
            width: 330
            height: 250
            radius: 30
            color: "dimgray"

            // ...
        }
    }
}
```

5. Кнопки покупки/продажи коинов.

```
Row {
    anchors.right: parent.right
    anchors.verticalCenter: parent.verticalCenter
    spacing: 10
    Button {
        anchors.verticalCenter: parent.verticalCenter
        width: 60
        text: "+"
        color: "white"
        onClicked: {
            var amount = parseInt(txtField.text)
            if (amount > 0) {
```

```
                    wallet[index + 1].sum += amount
                    wallet[0].sum -= amount * modelData.price
                    wallet = JSON.parse(JSON.stringify(wallet))
                }
            }
        }

        Button {
            anchors.verticalCenter: parent.verticalCenter
            width: 60
            text: "-"
            color: "white"
            onClicked: {
                var amount = parseInt(txtField.text)
                if (amount > 0 && wallet[index + 1].sum >= amount) {
                    wallet[index + 1].sum -= amount
                    wallet[0].sum += amount * modelData.price
                    wallet = JSON.parse(JSON.stringify(wallet))
                }
            }
        }
    }
```

6. Таймер управления поведением цены коина.

```
    Timer {
        running: true
        interval: Math.random() * 3000 + 1000
        repeat: true
        onTriggered: {
            var diff = (Math.random() - 0.5) / 100 + 1
            console.log(diff)
            coins[index].lastPrice = coins[index].price
            coins[index].price *= diff * 100
            coins[index].price = Math.round(coins[index].price) / 100

            coins = JSON.parse(JSON.stringify(coins))
        }
    }
```

7. Функция подсчета разницы прошлой и актуальной цены коина в процентах.

```
    function getDifference(price, lastPrice) {
        var diff = price / lastPrice * 100 - 100
        diff = diff * 100
        diff = Math.round(diff) / 100
        if (diff >= 0) diff = "+" + diff
        diff += "%"
        return diff
    }
```

Руководство пользователя

После запуска программы пользователь видит интерфейс с состоянием его кошелька, значением заработка, полем для ввода числа покупки/продажи и элементами, отражающими поведение цен коинов и кнопками покупки/продажи. Стартовый капитал пользователя – $10000.

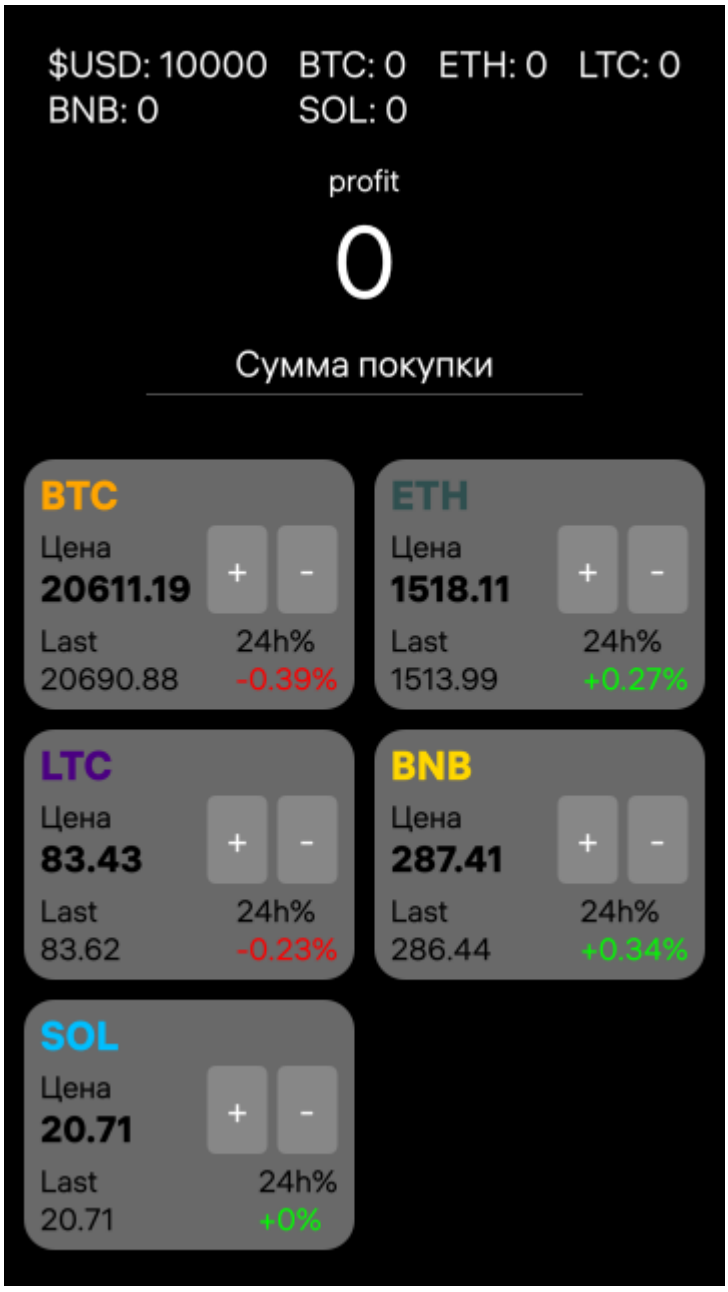

Рис. 1: Страница игры

Задача игры – следить за поведением цен на коины, делать соответствующие покупки и продажи так, чтобы максимизировать заработок.

Рис. 2: Процесс игры

Заключение

В процессе написания этого проекта мы успешно реализовали функционал игры «Крипто-биржа» – разработали поведение изменения цен на коины и интерфейс покупки и продажи.

Приложение

MainPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    PageHeader {
        objectName: ""
        extraContent.children: [
            IconButton {
                objectName: "aboutButton"
                icon.source: "image://theme/icon-m-capslock"
                anchors.verticalCenter: parent.verticalCenter

                onClicked: {
                    pageStack.clear()
                    pageStack.replace(Qt.resolvedUrl("../MainPage.qml"))
                }
            }
        ]
    }

    property int startingAmount: 10000

    property var coins: [
        {
            "name": "BTC",
            "lastPrice": 20690.88,
            "price": 20690.88,
            "color": "orange",
        },
        {
            "name": "ETH",
            "lastPrice": 1517.35,
            "price": 1517.35,
            "color": "darkslategray",
        },
        {
            "name": "LTC",
            "lastPrice": 83.56,
            "price": 83.56,
            "color": "indigo",
        },
        {
            "name": "BNB",
```

```
            "lastPrice": 286.59,
            "price": 286.59,
            "color": "gold",
        },
        {
            "name": "SOL",
            "lastPrice": 20.78,
            "price": 20.78,
            "color": "deepskyblue",
        },
    ]

    property var wallet: [
        {
            "name": "$USD",
            "sum": startingAmount,
        },
        {
            "name": "BTC",
            "sum": 0,
        },
        {
            "name": "ETH",
            "sum": 0,
        },
        {
            "name": "LTC",
            "sum": 0,
        },
        {
            "name": "BNB",
            "sum": 0,
        },
        {
            "name": "SOL",
            "sum": 0,
        },
    ]

    Rectangle {
        anchors.fill: parent
        color: "black"
    }


    Column {
        anchors.centerIn: parent
        spacing: 10

        Grid {
```

```
        anchors.horizontalCenter: parent.horizontalCenter
        columns: 4
        columnSpacing: 30

        Repeater {
            model: wallet
            delegate: Label {
                text: modelData.name + ": " + Math.round(modelData.sum * 100) / 10
                color: "white"
            }
        }
    }

    Item {
        width: 10
        height: 10
    }

    Column {
        anchors.horizontalCenter: parent.horizontalCenter
        Label {
            anchors.horizontalCenter: parent.horizontalCenter
            text: "profit"
            font.pixelSize: 30
            color: "white"
        }
        Label {
            anchors.horizontalCenter: parent.horizontalCenter
            text: {
                var total = 0.0
                total += wallet[0].sum
                for (var i = 1; i < wallet.length; i++) {
                    total += coins[i - 1].price * wallet[i].sum
                }
                total = Math.round((total - startingAmount) * 100) / 100
                return total
            }

            font.pixelSize: 100
            color: "white"
        }
    }

    TextField {
        id: txtField
        anchors.horizontalCenter: parent.horizontalCenter
        width: 500
        color: "white"
        placeholderText: "Сумма покупки"
        placeholderColor: "white"
```

```
    horizontalAlignment: Text.AlignHCenter
}

Grid {
    anchors.horizontalCenter: parent.horizontalCenter
    columns: 2
    spacing: 20
    Repeater {
        id: repeater
        model: coins
        delegate: Rectangle {
            width: 330
            height: 250
            radius: 30
            color: "dimgray"


            Column {

                anchors.centerIn: parent
                width: parent.width * 0.9

                Label {
                    text: modelData.name
                    font.pixelSize: 40
                    font.bold: true
                    color: modelData.color
                }

                Item {
                    height: 100
                    width: parent.width
                    Column {
                        anchors.verticalCenter: parent.verticalCenter
                        Label {
                            font.pixelSize: 30
                            text: "Цена"
                        }
                        Label {
                            font.bold: true
                            text: modelData.price
                        }
                    }

                    Row {
                        anchors.right: parent.right
                        anchors.verticalCenter: parent.verticalCenter
                        spacing: 10
                        Button {
                            anchors.verticalCenter: parent.verticalCenter
```

```
                width: 60
                text: "+"
                color: "white"
                onClicked: {
                    var amount = parseInt(txtField.text)
                    if (amount > 0) {
                        wallet[index + 1].sum += amount
                        wallet[0].sum -= amount * modelData.price
                        wallet = JSON.parse(JSON.stringify(wallet)
                    }
                }
            }

            Button {
                anchors.verticalCenter: parent.verticalCenter
                width: 60
                text: "-"
                color: "white"
                onClicked: {
                    var amount = parseInt(txtField.text)
                    if (amount > 0 && wallet[index + 1].sum >= amo
                        wallet[index + 1].sum -= amount
                        wallet[0].sum += amount * modelData.price
                        wallet = JSON.parse(JSON.stringify(wallet)
                    }
                }
            }
        }
    }

    Item {
        width: parent.width
        height: 80
        Column {
            anchors.verticalCenter: parent.verticalCenter
            Label {
                text: "Last"
                font.pixelSize: 30
            }
            Label {
                text: modelData.lastPrice
                font.pixelSize: 30
            }
        }

        Column {
            anchors.verticalCenter: parent.verticalCenter
            anchors.right: parent.right
            Label {
                text: "24h%"
```

```
                                font.pixelSize: 30
                            }
                            Label {
                                color: {
                                    if (text[0] == '+') return "lime"
                                    return "red"
                                }

                                text: getDifference(modelData.price, modelData.las
                                font.pixelSize: 30
                            }
                        }
                    }
                }

                Timer {
                    running: true
                    interval: Math.random() * 3000 + 1000
                    repeat: true
                    onTriggered: {
                        var diff = (Math.random() - 0.5) / 100 + 1
                        console.log(diff)
                        coins[index].lastPrice = coins[index].price
                        coins[index].price *= diff * 100
                        coins[index].price = Math.round(coins[index].price) / 100

                        coins = JSON.parse(JSON.stringify(coins))
                    }
                }
            }
        }
    }

    function getDifference(price, lastPrice) {
        var diff = price / lastPrice * 100 - 100
        diff = diff * 100
        diff = Math.round(diff) / 100
        if (diff >= 0) diff = "+" + diff
        diff += "%"
        return diff
    }
}
```