МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровой экономики

КУРСОВОЙ ПРОЕКТ

По дисциплине «Проектная деятельность»

«Разработка игры в жанре текстовый симулятор жизни с помощью технологии NativeScript для разработки мобильных приложений»

Выполнил: студент 2 курса 1192б группы

Доронин Всеволод Михайлович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель:

Сафонов Егор Иванович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

г. Ханты-Мансийск

2021 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc73328532)

[2 ВЫБОР ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ 4](#_Toc73328533)

[2.1. JavaScript 4](#_Toc73328534)

[2.2. Vue.js 4](#_Toc73328535)

[2.3 NativeScript 4](#_Toc73328536)

[2.3 Visual Studio Code 5](#_Toc73328537)

[3 РАЗРАБОТКА ПРИЛОЖЕНИЯ 6](#_Toc73328538)

[3.1 Vue компоненты приложения. 6](#_Toc73328539)

[3.2 Разметка экрана 6](#_Toc73328540)

[3.3 Применение стилей 6](#_Toc73328541)

[ЗАКЛЮЧЕНИЕ 7](#_Toc73328542)

[СПИСОК ЛИТЕРАТУРЫ 8](#_Toc73328543)

[ПРИЛОЖЕНИЕ А 9](#_Toc73328544)

[ПРИЛОЖЕНИЕ Б 10](#_Toc73328545)

# ВВЕДЕНИЕ

Данная работа основывается на понятии «мобильная разработка». Это процесс, при котором приложения разрабатываются для небольших портативных устройств, таких, как КПК, смартфоны или сотовые телефоны. Эти приложения могут быть предустановлены на устройство в процессе производства, загружены пользователем с помощью различных платформ для распространения ПО или являться веб-приложениями, которые обрабатываются на стороне клиента или сервера.

Наиболее распространенным способом разработки приложений под ОС Android является IDE Android Studio, официально поддерживаемая Google. Разработка ведется на языке программирования Java или Kotlin. [1]

# 2 ВЫБОР ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ

Для разработки был выбран следующий набор инструментов: язык программирования JavaScript, фреймворк для создания пользовательских интерфейсов Vue.js и фреймворк для мобильной разработки NativeScript. В качестве среды разработки выступал редактор кода Visual Studio Code.

## 2.1. JavaScript

JavaScript изначально являлся скриптовым языком программирования, чтобы сделать веб-страницы «живыми». Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя. [2]

## 2.2. Vue.js

Vue.js это реактивный фреймворк для JavaScript. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений (SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками. [3]

## 2.3 NativeScript

NativeScript - это основа для создания кросс-платформенных нативных мобильных приложений. Она позволяет разработчикам использовать XML, CSS и JavaScript для создания приложений для Android, iOS и даже для Windows Universal Platform. В отличие от Cordova, которая использует WebView для визуализации пользовательского интерфейса приложения, NativeScript использует механизм рендеринга собственной платформы, что означает, что он обеспечивает действительно собственный пользовательский интерфейс. В работе в приложением применялся плагин NativeScript-Vue для NativeScript, который дает возможность ипспользовать Vue.js для создания мобильного приложения. [4][5]

## 2.3 Visual Studio Code

Visual Studio Code – это один из наиболее популярных редакторов кода, разработанный корпорацией Microsoft. Он распространяется в бесплатном доступе и поддерживается всеми актуальными операционными системами: Windows, Linux и macOS. VS Code представляет собой обычный текстовый редактор с возможностью подключения различных плагинов, что дает возможность работать со всевозможными языками программирования для разработки любого IT-продукта. [6]

Несмотря на столь высокую популярность программы, ее функционал до сих пор остается не таким очевидным, из-за чего многие пользователи предпочитают продукт конкурента, нежели VS Code.

# 3 РАЗРАБОТКА ПРИЛОЖЕНИЯ

Основными задачами данного приложения были ознакомление с технологиями мобильной разработки и нетривиальное применение данных технологий. Идея приложения – игра в жанре текстовый симулятор жизни, подобная тем, что были раньше. Реализация интерфейса продемонстрирована в Приложении А в Рисунке А.1.

## 3.1 Vue компоненты приложения

Приложение, в целях декомпозиции элементов, было разбито на несколько компонентов Vue, которые отвечают за свои задачи и содержат свой функционал. Компоненты это переиспользуемые экземпляры Vue, то есть они принимают опции data, computed, watch, methods, хуки жизненного цикла. Структура компонентов в проекте древовидна, как изначально и подразумевалось.

Главный компонент, запускающийся при старте программы, содержит разметку нескольких кнопок, скриптов и путь к стилю, который применяется и в дочерних компонентах. Другие компоненты вызываются с помощью функции this.$showModal(Game, {fullscreen: true,}), показывающая модальное окно. Стандартный компонент App.vue представлен в Листинге кода Б.1 (Приложение Б).

## 3.2 Разметка экрана

Разметка экрана в NativeScript-Vue не отличается от разметки экрана в верстке страниц сайта. В отличие от стандартной HTML разметки, в NS-Vue используется XML разметка со специфичными для мобильной разработки тэгами типа <Button/>, <Label/>, <GridLayout/>. Структура и пример разметки представлены в Листинге кода Б.2 (Приложение Б).

В компонентах Vue разметка, скрипт и стили разделены между собой разными тэгами. Содержимому страницы соответствует тэг <template> (шаблон), в котором и находятся тэги.

## 3.3 Применение стилей

При мобильной разработке с помощью NativeScript-Vue используются каскадные таблицы стилей (CSS) и их производные. При создании проекта по умолчанию создается глобальный файл стиля в формате .SCSS. Он был заменен файлом style.css, содержащий стандартные стили, интегрированные в XML разметку. Код style.css представлен в Листинге кода Б.3 (Приложение Б).

## 3.4 Хранилище данных Vuex

В целях создания единого хранилища данных и декомпозиции приложения был использован Vuex – паттерн управления состоянием + библиотека для приложений на Vue.js.

Концепция Vuex подразумевает собой перенос всей логики (переменных, методов) в отдельный скрипт хранилища. В изначальном варианте приложения хранилище не было нужным, так как весь код находился в одном Vue компоненте. По мере увеличения программы появилась необходимость разделить весь код на несколько файлов: компоненты, хранилище. Файл store.js в свою очередь тоже можно декомпозировать, разбив на несколько модулей, что в данном проекте было целесообразно, но не являлось приоритетной задачей. Код данного файла представлен в Листинге кода Б.4 (Приложение Б).

## 3.5 Функционал приложения

Приложение – текстовая игра в жанре симулятор жизни. Идея приложения нетривиальна с учетом назначения использованных инструментов. Такой вариант приложения был выбран по ряду причин. Во-первых, простота программирования логики – в основе всего лежат несколько однотипных функций. Во-вторых, наличие большого количества примеров.

При запуске приложения появляется «главное меню», содержащее несколько кнопок: «новая игра», «продолжить», «загрузка», «настройки», «о», «выход». При нажатии на первые три запускается игра, при нажатии на «настройки» открывается модальное окно с настройками, при нажатии на «о» открывается предупреждающее окошко с надписью «о» (подразумевается информация о разработчике), при нажатии на «выход» приложение закрывается.

Функционал настроек еще не создан. В концепции подразумевается, что из этой вкладки будет происходить манипуляция со стартовыми параметрами персонажа, такими как возраст, начальные параметры и т.д.

В самой игре интерфейс простой – экран разделенный на несколько вкладок (TabView). Во вкладках кнопки отвечают за изменение разных переменных. Во вкладке «работа» кнопки уменьшают параметры «голод», «здоровье» и «настроение», но увеличивают параметр «рубли». Во вкладке «потребности» ситуация обратная.

Особенностью вкладки «работа» является неактивность некоторым элементов, пока параметры, связанные с образованием не станут истинными. То есть пока персонаж не получит параметр edTierTwo = true кнопки «мали работа» не станет активна.

# ЗАКЛЮЧЕНИЕ

В данном проекте был получен опыт мобильной разработки и подготовлено приложение, которое потенциально может быть загружено в AppStore и PlayMarket.

# СПИСОК ЛИТЕРАТУРЫ

1. Android Developers [электронный ресурс] / Download Android Studio and SDK tools  |  Android Developers / режим доступа - . - <https://developer.android.com/studio?hl=sl>, свободный (дата обращения 31.05.2021)
2. Javascript.ru [электронный ресурс] / Современный учебник JavaScript / 2021, режим доступа - . - <https://learn.javascript.ru/>, свободный (дата обращения: 31.05.2021)
3. Vue.js [электронный ресурс] / Vue.js / 2021 - . - режим доступа <https://ru.vuejs.org/>, свободный (дата обращения: 31.05.2021)
4. NativeScript [электронный ресурс] / NativeScript / 2021 - . - режим доступа <https://nativescript.org/>, свободный (дата обращения: 31.05.2021)
5. NativeScript-Vue [электронный ресурс] / NativeScript-Vue / 2021 - . - режим доступа <https://nativescript-vue.org/ru/>, свободный (дата обращения: 31.05.2021)
6. комьюнити. [электронный ресурс] / Как пользоваться Visual Studio Code / 2021 - . - режим доступа <https://timeweb.com/ru/community/articles/chto-takoe-visual-studio-code>, свободный (дата обращения: 31.05.2021)

# ПРИЛОЖЕНИЕ А

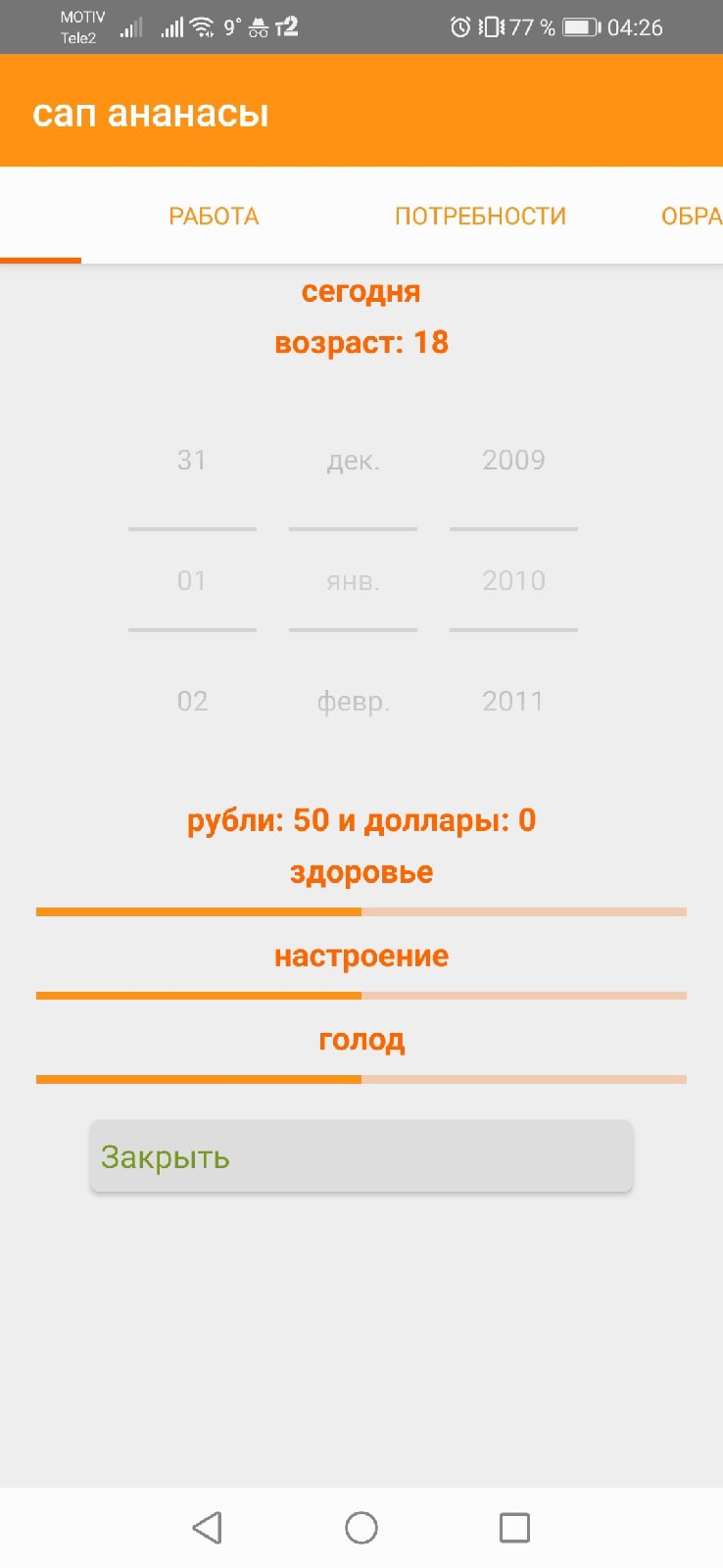


Рисунок А.1 – «Интерфейс приложения»

# ПРИЛОЖЕНИЕ Б

Листинг кода Б.1. – Компонент App.vue

<template>

<Page>

<ActionBar title="добро пожаловать. снова."/>

<StackLayout>

<Button class="button" @tap="startNewGame(), newGameNormal()" text=">новая игра"/>

<Button class="button" @tap="startLastSave()" text=">продолжить"/>

<Button class="button" @tap="starAnySave()" text=">загрузить игру"/>

<Button class="button" @tap="settings()" text=">настройки"/>

<Button class="button" @tap="about()" text=">о"/>

<Button class="button" @tap="closeApp()" text=">выход"/>

</StackLayout>

</Page>

</template>

<script >

import {mapMutations} from 'vuex';

import Game from './Game';

import Settings from './Settings';

import \* as application from "@nativescript/core/application";

  export default {

    methods:{

      ...mapMutations(['newGameNormal']),

      startNewGame: function(*event*){

        this.$showModal(Game, {

          fullscreen: true,

        })

      },

      startLastSave: function(*event*){

        this.$showModal(Game, {

          fullscreen: true,

        })

      },

      starAnySave: function(*event*){

        this.$showModal(Game, {

          fullscreen: true,

        })

      },

      settings: function(*event*){

        this.$showModal(Settings,{

          fullscreen: true,

        })

      },

      about() {

        alert(({

          title: "о",

          button: "о",

          okButtonText: "о"

        }));

      },

      closeApp() {

        if (application.android) {

          application.android.foregroundActivity.finish();

        } else {

          exit(0);

        }

      }

    }

  }

</script>

<style src="./style.css"></style>

Листинг кода Б.2 – Пример разметки компонента

<template>

<GridLayout columns='\*' rows='30, 50, auto'>

  <label row='0' class="big-text">ЗА РАБОТУ</label>

  <GridLayout row='1' columns='\*, \*, \*' rows='20, 30'>

    <label class='text' col='0' row='0'>голод ({{hunger}}/100)</label>

    <Progress width='30%' col='0' row='1' :value="hunger" maxValue="100"/>

    <label class='text' col='1' row='0'>настроение ({{mood}}/100)</label>

    <Progress width='30%' col='1' row='1' :value="mood" maxValue="100" />

    <label class='text' col='3' row='0'>здоровье ({{health}}/100)</label>

    <Progress width='30%' col='2' row='1' :value="health" maxValue="100"/>

  </GridLayout>

  <StackLayout v-if="hunger>0 || health>0 || mood>0" row='2' width='' orientation='vertical'>

    <Button class="button" text=">оче мали работа (+100р))" @tap="workTierOne()"/>

    <Button v-if='edTierTwoGtr' class="button" text=">мали работа (+500р)" @tap="workTierTwo()"/>

    <Button v-else isEnabled='false' class="button" text=">мали работа (+500р)" @tap="workTierTwo()"/>

    <Button v-if='edTierThreeGtr' class="button" text=">работа (+1000р)" @tap="workTierThree()"/>

    <Button v-else isEnabled='false' class="button" text=">работа (+1000р)" @tap="workTierThree()"/>

    <Button v-if='edTierFourGtr' class="button" text=">больче работа (+5000р)" @tap="workTierFour()"/>

    <Button v-else isEnabled='false' class="button" text=">больче работа (+5000р)" @tap="workTierFour()" />

    <Button v-if='edTierFiveGtr' class="button" text=">оче больче работа (+1000р)" @tap="workTierFive()"/>

    <Button v-else isEnabled='false' class="button" text=">оче больче работа (+10000р)" @tap="workTierFive()"/>

  </StackLayout>

  <Label textWrap='true' row='2' v-else class="big-text" text='видимо, какой-то из параметров упал слишком настолько низко, что смерть настигла сего аватара'/>

</GridLayout>

</template>

Листинг кода Б.3 – Стили

ActionBar {

    background-color: #ff9113;

    color: #fefefe;

}

Page {

    background-color: #eeeeee;

}

.button {

    vertical-align: center;

    text-align: left;

    color: #789922;

    background-color: #dddddd;

    font-size: 16px;

    font-family: sans-serif;

    width: 75%;

    height: 36rem;

    margin-top: 12rem;

    border-radius: 5%;

}

Progress {

    background-color: #ff6600;

    color: #ff9113;

    width: 90%;

}

TabView{

    tab-background-color: #fcfcfc;

    tab-text-color: #ff9113;

    selected-tab-text-color: #ff6600;

    android-selected-tab-highlight-color: #ff6600;

}

.text {

    text-align: center;

    color: #333333;

    font-size: 12px;

    font-family: sans-serif;

}

.big-text {

    text-align: center;

    color: #ff6600;

    font-size: 16px;

    font-family: Impact;

    font-weight: bold;

}

Листинг кода Б.4 – Хранилище данных store.js

import Vue from 'vue';

import Vuex from 'vuex';

const Toast = require('nativescript-toast');

import \* as application from "@nativescript/core/application";

Vue.use(Vuex);

export default new Vuex.Store({

  state: {

    character: {

      params: {

        age: 18,

        health: 50,

        mood: 50,

        hunger: 50,

      },

      money:{

        rub: 50,

      },

      education: {

        tierTwo: false,

        tierThree: false,

        tierFour: false,

        tierFive: false,

      },

      estate:{

        tierTwo: false,

        tierThree: false,

        tierFour: false,

        tierFive: false,

      },

      transport: {

        tierTwo: false,

        tierThree: false,

        tierFour: false,

        tierFive: false,

      },

    },

    date: {

      y: 2010,

      m: 1,

      d: 1,

    }

  },

  getters: {

    rub(*state*) {

      return state.character.money.rub

    },

    health(*state*) {

      return state.character.params.health

    },

    hunger(*state*) {

      return state.character.params.hunger

    },

    mood(*state*) {

      return state.character.params.mood

    },

    age(*state*){

      return state.character.params.age

    },

    d(*state*){

      return state.date.d

    },

    m(*state*){

      return state.date.m

    },

    y(*state*){

      return state.date.y

    },

    edTierTwoGtr(*state*){

      return state.character.education.tierTwo

    },

}

 mutations: {

*/\* ненормальные мутации,где меняются все данные в мутации - отвечают за уровни сложности (если сделаю)) \*/*

    newGameNormal(*state*){

      state.character.params.age = 18;

      state.character.params.health = 50;

      state.character.params.hunger = 50;

      state.character.params.mood = 50;

      state.character.money.rub = 50;

      state.character.money.usd = 0;

      state.character.education.tierTwo = false;

      state.character.education.tierThree = false;

      state.character.education.tierFour = false;

      state.character.education.tierFive = false;

      state.character.transport.tierTwo = false;

      state.character.transport.tierThree = false;

      state.character.transport.tierFour = false;

      state.character.transport.tierFive = false;

      state.character.estate.tierTwo = false;

      state.character.estate.tierThree = false;

      state.character.estate.tierFour = false;

      state.character.estate.tierFive = false;

      state.date.y = 2010;

      state.date.m = 1;

      state.date.d = 1;

    },

    rub(*state*, *a*){

      state.character.money.rub += a

    },

    hunger(*state*, *a*){

      state.character.params.hunger += a

    },

    health(*state*, *a*){

      state.character.params.health += a

    },

    mood(*state*, *a*){

      state.character.params.mood += a

    },

    dayCounter(*state*){

      state.date.d +=1

      if((state.date.d > 31 && (state.date.m == 1 || state.date.m == 3 ||state.date.m == 5 ||state.date.m == 7 ||state.date.m == 8 ||state.date.m == 10 ||state.date.m == 12)) || (state.date.d > 30 && (state.date.m == 4 || state.date.m == 6 || state.date.m == 9 || state.date.m == 11)) || (state.date.m == 2 && ((state.date.d > 29 && state.date.y % 4 == 0) || (state.date.d > 28 && state.date.y %4 !=0 )))) {

        state.date.d = 1;

        state.date.m +=1;

        if(state.date.m > 12){

          state.date.m = 1;

          state.date.y +=1;

          state.character.params.age +=1;

        }

      }

    },

    edTierTwo(*state*){

      state.character.education.tierTwo = true

    },

    edTierThree(*state*){

      state.character.education.tierThree= true

    },

    edTierFour(*state*){

      state.character.education.tierFour = true

    },

    edTierFive(*state*){

      state.character.education.tierFive= true

    },

    esTierTwo(*state*){

      state.character.estate.tierTwo = true

    },

    esTierThree(*state*){

      state.character.estate.tierThree= true

    },

    esTierFour(*state*){

      state.character.estate.tierFour = true

    },

    esTierFive(*state*){

      state.character.estate.tierFive= true

    },

    trTierTwo(*state*){

      state.character.transport.tierTwo = true

    },

    trTierThree(*state*){

      state.character.transport.tierThree= true

    },

    trTierFour(*state*){

      state.character.transport.tierFour = true

    },

    trTierFive(*state*){

      state.character.transport.tierFive = true

    },

    alertRub(){

      alert(({

        title: "денях нет",

        button: "e",

        okButtonText: "ладно"

      }))

    },

    alertDeath(){

      alert(({

        title: "игра оконочена",

        button: "ок",

        okButtonText: "ок"

      }))

    }

  },

  actions: {

    workTierOne(*ctx*){

      if (ctx.state.character.params.health <= 0 || ctx.state.character.params.hunger <= 0|| ctx.state.character.params.mood <=0){

        ctx.commit('alertDeath');

        ctx.state.character.params.health = 0;

        ctx.state.character.params.hunger = 0;

        ctx.state.character.params.mood = 0;

      }

      else{

        ctx.commit('rub', 100);

        ctx.commit('health', -5);

        ctx.commit('hunger', -5);

        ctx.commit('mood', -5);

        ctx.commit('dayCounter');

      }

    },

},

});