**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Старший преподаватель департамента программной инженерии факультета компьютерных наук  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Максименкова  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**ИГРА "PIXEL WARS"**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.508830-01 81 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ173

**../../Снимок%20экрана%202018-05-21%20в%2010.23.26.png**/ С. И. Ройтман /

«17» мая 2018 г.

**Москва 2018**

**УТВЕРЖДЕН**

**RU.17701729.508830-01 81 01-1-ЛУ**

**ИГРА "PIXEL WARS"**

**Пояснительная записка**

**RU.17701729.508830-01 81 01-1**

**Листов 21**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2018**

**СОДЕРЖАНИЕ**

1. ВВЕДЕНИЕ 3

1.1. Наименование программы 3

1.2. Документы на основании которых ведётся разработка 3

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4

2.1. Функциональное назначение 4

2.2. Эксплуатационное назначение 4

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5

3.1. Постановка задачи на разработку программы 5

3.2. Описание функционирования программы 5

3.2.1 Серверная часть 5

3.2.2 Клиентская часть 5

3.3 Описание и обоснование метода организации входных и выходных данных 6

3.4 Описание и обоснование выбора технических и программных средств. 6

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 8

4.1. Ориентировочная экономическая эффективность 8

4.2. Предполагаемая потребность 8

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 8

5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 9

Приложение 1 «Диаграмма классов» 10

Приложение 2 «Описание и функц. назначение классов» 11

Приложение 3 «Описание и функц. назначение полей/свойств и методов классов» 11

1. **ВВЕДЕНИЕ**
   1. **Наименование программы**  
      Наименование программы: «Игра Pixel Wars», (Краткое название: «PW»).
   2. **Документы на основании которых ведется разработка**

Приказ Национального Исследовательского Университета «Высшая школа экономики» № 2.3-02/0812-01 от 08.12.2018. Программа выполняется в рамках темы курсовой работы, в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия» Национального исследовательского университета «Высшая школа экономики», факультет компьютерных наук, департамент программной инженерии.

1. **НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**
   1. **Функциональное назначение**  
      Программа представляет из себя онлайн 2D игру, в ней существует игровое поле размером 100\*100 пикселей. Все пиксели изначально белого цвета, но пользователи могут раз в минуту изменять цвет одного из пикселей на любой другой. Таким образом игра представляет собой своеобразный холст, который пользователи, разукрашивают, кооперируясь между собой, при этом им придется соревноваться с соперниками за свободное пространство.
   2. **Эксплуатационное назначение**  
      Приложение предназначено для пользователей разных возрастов на девайсах под управлением ОС Android. Для программы обязательно наличие стабильного Интернет подключения.
2. **ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**
   1. **Постановка задачи на разработку программы**

Основной задачей было создать многопользовательскую игру, в которой пользователи могли бы проявлять свои творческие способности и создавать пиксельные рисунки.

* 1. **Описание функционирования программы**  
      Программный продукт включает в себя использование нескольких языков программирования и технологий. На языке C# .NET была реализована серверная часть приложения, а на языке Java (с использованием таких библиотек как «Glide», «Colorpicker», «TouchImageView») в среде разработки Android Studio клиентская часть.
     1. Серверная часть состоит из 4 классов: Program, Server, Client и Tools.
     + **Класс Program**  
       Основная часть, необходимая для создания и запуска сервера.
     + **Класс Server**  
       Реализация методов взаимодействия по системе КЛИЕНТ-СЕРВЕР, а также все необходимые методы для обеспечения игровой деятельности игрока (согласно «Техническому заданию»).
     + **Класс Client**  
       Класс представляет собой образ клиентов, взаимодействующих с сервером.
     + **Класс Tools**  
       Методы общего назначения, необходимые для корректной работы приложения. Среди них — конвертация картинки в байты, XML сериализация и десериализация.
     1. Клиентская часть состоит из 7 классов:
     + **Класс EnterActivity**  
       Окно авторизации в игре.
     + **Класс RegistrationActivity**  
       Окно регистрации в системе.
     + **Класс GameActivity**  
       Основное окно, где происходит игровой процесс.
     + **Класс InternetConnection**  
       Реализация методов взаимодействия по системе КЛИЕНТ-СЕРВЕР: соединение с сервером, передача и получение сообщений.
     + **Класс Responder**

Обработчик нажатий на игровое поле.

* + - **Класс RefreshFieldTask**

Служба для подгружения поля с сервера в фоновом режиме.

* + - **Класс TouchImageView**

Импортированный класс для реализация игрового поля: скроллинга, зума и т.п.

* 1. **Описание и обоснование метода организации входных и выходных данных**  
      Входными данными является параметры, которые задаёт пользователь через интерфейс клиентской части приложения путем нажатия на элементы управления и ввод данных в соответствующие поля.

Данные передаются на сервер как поток байт, а затем дешифруются. Таким же образом происходит и обратная передача информации (от сервера к клиенту). Данная технология передачи данных была выбрана ввиду практичности и простоты такого метода.

* 1. **Описание и обоснование выбора технических и программных средств**  
      Языком серверной части был выбран **C# .NET**, ввиду его изучения во время учебного процесса.

Для реализации клиентской части приложения была выбрана среда разработки Android Studio и язык программирования Java, т.к. эти технологии являются стандартом при разработке мобильных приложений.  
 Для работы программного продукта требуется следующий состав программных и технических средств:  
**Клиентская часть:**

– Рекомендуется смартфон под управлением ОС Android версии 4.0.3 или выше

– Для работы приложения необходимо стабильное интернет соединение.  
**Серверная часть:**  
– Рекомендуется 1024 МБ ОЗУ или более на 1000 подключенных пользователей приложения (далее — линейная зависимость);

– Не менее 10 МБ свободного места на жёстком диске или носителе, на котором исполняется программа;

– Интернет-канал со скоростью соединения 130 Мбит/сек или более на 1000 подключенных пользователей приложения (далее — линейная зависимость);  
– Windows Server 2012 R2 или выше;

– Режим постоянной работы сервера (не переходя в спящий или аналогичный режим при неактивности);

1. **ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**
   1. **Ориентировочная экономическая эффективность** В рамках данной работы расчёт экономической эффективности не предусмотрен.
   2. **Предполагаемая потребность** Данный продукт предназначен для развлечения и развития творческих способностей.
   3. **Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами** В рамках данной работы расчёт экономической эффективности не предусмотрен.
2. **ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**
3. Опыт применения ЕСПД // @KirillAlexandrovich. – М.: Издательство интернет-проектов TechMedia / «Хабрахабр», 2014.
4. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. Android. Программирование для профессионалов. //Билл Филлипс, К. Стюарт, Кристин Марсикано. – М.: Издательство Питер, 2017.
6. Консольный TCP-чат// М.: Интернет-издательство Metanit.com, 2015.
7. How to implement IXmlSerializable correctly // Jaap de Haan – М.: Издательство интернет-проектов CodeProject, 2009.
8. Многопоточный сервер на C# за 15 минут// @ertaquo. – М.: Издательство интернет-проектов TechMedia / «Хабрахабр», 2009.
9. Простой клиент-сервер на Android (интернет-мессенджер)// @ andreidanilevich. – М.: Издательство интернет-проектов TechMedia / «Хабрахабр», 2015.
10. ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

**Приложение 1**

**Диаграмма классов**

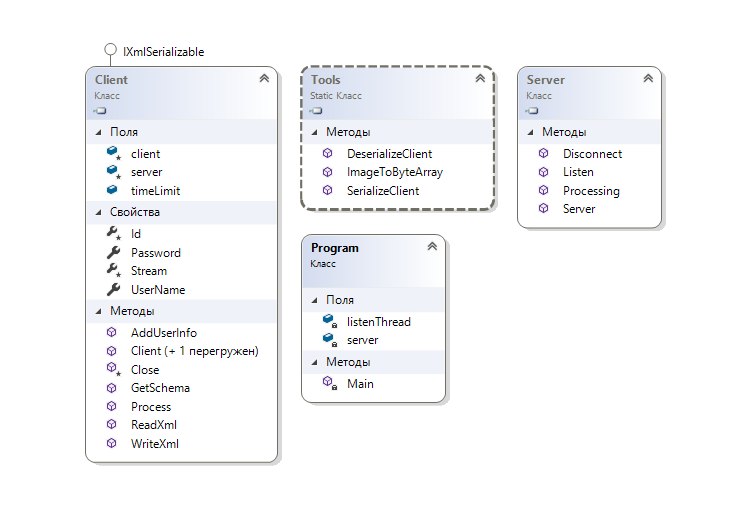


Рис. 1 / Диаграмма классов

**Приложение 2**

**Описание и функциональное назначение классов**

**C# .NET**

|  |  |
| --- | --- |
| **Имя класса** | **Назначение** |
| *Program* | Основная часть, необходимая для создания и запуска сервера. |
| *Server* | Реализация методов взаимодействия по системе КЛИЕНТ-СЕРВЕР, а также все необходимые методы для обеспечения игровой деятельности игрока (согласно «Техническому заданию»). |
| Client | Класс представляет собой образ клиентов, взаимодействующих с сервером. |
| Tools | Методы общего назначения, необходимые для корректной работы приложения. Среди них — конвертация картинки в байты, XML сериализация и десериализация. |

**Приложение 3**

**Описание и функциональное назначение полей/свойств и методов классов**

**C# .NET**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Класс Program** | | | | | | | |
| **Поля и свойства** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Описание | |
| server | private | | Server | | | Сервер | |
| listenThread | private | | Thread | | | Поток для прослушивания | |
| **Методы** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Параметры | | Описание |
| Main | | private | | void | string[] args | | Создает сервер, запускает его. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс *Server*** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| tcpListener | private | TcpListener | Сервер для прослушивания | |
| clients | private | List<Client> | Список подключенных игроков | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| Server | public | Конструктор |  | Создает сервер и игровое поле. |
| Processing | public | void | id, msg | Обрабатывает запросы клиентов. |
| CreateField | private | void |  | Рисует начальное игровое поле и сохраняет его в виде картинки. |
| SetPixelColor | private | void | x, y, color | Раскрашивает пиксель(x, y) в цвет color и сохраняет поле. |
| LoginUser | private | bool | nickname, password | Возвращает true, если данный пользователь зарегистрирован. |
| UpdateUserTimeLimit | private | void | id | Обновляет временое ограничение пользователя на изменение цвета пикселей. |
| GetTimeLimit | private | int | id | Возвращает количество секунд прошедшее со времени последнего изменения цвета пользователем. |
| SendMessageToUser | private | void | message, id | Отправка сообщения клиенту. |
| SendFieldToUser | private | void | id | Отправка текущего состояния поля клиенту. |
| Listen | public | void |  | Прослушивание входящих соединений. |
| AddConnection | internal | void | client | Добавляет клиента. |
| RemoveConnection | internal | void | id | Удаляет клиента. |
| Disconnect | public | void |  | Отключает всех клиентов. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Client** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| Id | protected internal | ind | ID игрока, равен номеру в списке игроков. | |
| Stream | protected internal | NetworkStream | Поток для доступа к сети. | |
| UserName | public | string | Свойство для чтения никнейма игрока. | |
| userName | private | string | Никнейм игрока. | |
| Password | public | string | Свойство для чтения пароля пользователя. | |
| password | private | string | Пароль пользователя. | |
| timeLimit | Public | DateTime | Дата последнего действия(изменения цвета пикселя). | |
| client | protected internal | TcpClient | Предоставляет клиентские подключения для сетевых служб протокола TCP. | |
| server | protected internal | Server | Объект сервера. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| Client | public | Конструктор | tcpClient, server | Создает клиента, добавляет его в список клиентов. |
| WriteXml | public | void | writer | Реализация интерфейса IXmlSerializable. |
| ReadXml | public | void | reader | Реализация интерфейса IXmlSerializable. |
| GetSchema | public | XmlSchema |  | Реализация интерфейса IXmlSerializable. |
| AddUserInfo | public | void | userName, password | Добавляет информацию о пользователе. |
| Process | public | void |  | Ожидание и получение сообщений от клиента. |
| GetMessage | private | string |  | Чтение сообщения от клиента и преобразование его в строку. |
| Close | Protected internal | void |  | Закрытие подключения. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Tools** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| ImageToByteArray | public | byte[] | image, format | Преобразует картинку в набор байт. |
| SerializeClient | public | void | client | Создает XML файл с данными пользователя на сервере. |
| DeserializeClient | public | Client | userName | Расшифровывает XML файл с пользователя с данным никнеймом и возвращает клиента. |

**Java, Android Studio**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Класс EnterActivity** | | | | | |
| **Методы** | | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| onCreate | protected | void | savedInstanceState | Выполняется при открытии приложения, соединяется с сервером. |
| login | public | void | view | Выполняется при нажатии кнопки “Login”. Посылает на сервер введенные пользователем логин и пароль для входа в игру, получает ответ. |
| registration | public | void | view | Выполняется при нажатии кнопки “Register”. Переходит на страницу регистрации. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Класс RegistrationActivity** | | | | | |
| **Методы** | | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| onCreate | protected | void | savedInstanceState | Выполняется при переходе на окно регистрации. |
| register\_user | public | void | view | Выполняется при нажатии кнопки “Register”. Проверяет и посылает на сервер введенные пользователем логин и пароль, получает ответ. |
| CheckPattern | public | boolean | s | Проверяет состоит ли строка только из букв и цифр. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Класс GameActivity** | | | | | |
| **Методы** | | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| onCreate | protected | void | savedInstanceState | Выполняется при переходе на окно с игровым полем.  Получает от сервера поле, создает обработчик нажатий по полю, запрашивает у сервера время предыдущего действия игрока, запускает таймер, запрашивающий новое поле раз в 10 секунд. |
| getTimeLimit | public | void |  | Получает время предыдущего действия игрока и запускает таймер, ограничивающий действия игрока. |
| StartRefreshFieldTimer | public | void | seconds | Запускает таймер, запрашивающий у сервера поле раз в seconds секунд. |
| startTimer | public | void | seconds | Запускает таймер на seconds секунд, ограничивающий действия игрока по изменению цвета пикселей. |
| RefreshField | public | void | data | Обновляет игровое поле: загружает картинку, полученную из набора байт. |
| ShowColourPicker | public | void | cell\_x, cell\_y | Создает меню для выбора цвета пикселя, когда цвет выбран отсылает на сервер запрос на изменения цвета пикселя (cell\_x, cell\_y). |
| StartRefreshFieldTimer | public | void | seconds | Запускает таймер, запрашивающий у сервера поле раз в seconds секунд. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс InternetConnection** | | | | |
| **Поля и свойства** | | | | |
| Имя | Модификатор доступа | Тип | Описание | |
| inFromUser | public | BufferedReader | Входной поток. | |
| inFromServer | public | DataInputStream | Поток для чтения с сервера. | |
| outToServer | public | DataOutputStream | Поток для отправки на сервер. | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| connectToServer | public | String |  | Устанавливает соединение с сервером. |
| sendMessageToServer | public | void | msg | Отправляет бинарно-сериализованное сообщение на сервер. |
| getMessageFromServer | public | String |  | Возвращает десериализованное сообщение с сервера. |
| getFieldFromServer | public | byte[] |  | Посылает на сервер запрос, а затем возвращает поток байт, в котором содержится текущее игровое поле. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Responder** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| singleTapHandler | public | String | x, y | Обработчик нажатия на игровое поле. Если ограничених на действия игрока нет, то открывает меню выбора цвета пикселя (x, y). |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс RefreshFieldTask** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| doInBackground | public | byte[] | Void… params | Служба для подгружения поля в фоновом режиме. |
| onPostExecute | public | void | result | Возвращает подгруженное поле. |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |