***Описание документа.***

Мы будем выделять отдельные части проекта(игры), т.к. нельзя просто взять и сделать игру. А, нужно «кристаллизовать» процесс, т.е. разбить его на части, задачи и этапы.

Игра будет выглядеть как на скриншотах, я их сюда пока что не прицепил. Прицеплю как нить в следующих версиях, в т.ч. отдельные скриншоты для этапов…

Весь проект(игра) состоит из отдельных частей:

- **сервер**. Это нужно что бы игроки могли обмениваться друг с другом данными, так уж устроен интернет. Для хранения профилей игроков (очки, сыгранные игры и т.д.);

- **клиент.** Это та - самая игра которая открывается в браузере, когда мы заходим, например, в игру во вКонтакте.

- **графика**. Тут понятно, мы должны понимать, что графика рисуется отдельно от программирования, тут всё просто вроде.

- **бизнес-логика**. Т.е. её механизмы всякие, идеи и прочие, например, «Игрок может отправить приглашение другому пользователю» - это логика, сюда не входит графика, и даже программный код, а только именно процесс, логика, описанная на бумаге или в электронном документе. «Научно» это называется **бизнес-логика**, от английского business-logic (дело и логика).

Далее будет описан **План,** в конце плана будет готовый продукт, которым можно пользоваться и радоваться.

План мы разделим на этапы. **Этап** – это будет набор того, что нужно сделать, что бы можно было выполнить какую-то отдельную, целостную часть проекта(игры). Например, создание чата в игре, мы можем описать несколькими задачами, но все их мы включим в ОДИН этап, так, что если этап будет выполнен, то мы получим готовый, рабочий чат, но не больше того. Т.е. мы не будем включать в этап, ничего кроме чата.

Надо сказать, что в процессе разработки, каждый новый этап будет сталкиваться с тем, что сделанное ранее не работает, и надо будет чинить, это нормальная практика.

Каждый этап будет описан в следующем виде.

Этап №: **Название Этапа**

**Результат:**

Конечный результат этапа. Что мы сможем увидеть, потрогать, потыкать…

*Задачи:*

№ - Называние задачи – описывает, что делаем. (*Код – кодовое название)*

Тут описание - зачем делается эта задача, к чему мы придем и что получим, что можно будет увидеть, потыкать.

Этап 1. **Базовый функционал.**

**Результат:** результата внешне никакого☺ пустой экран☺ это всё программистская магия.  
Просто куча файлов с кодом☺ Но всё это нужно для реализации дальнейших этапов.

*Задачи:*

1 - Функционал для работы с базой данных. (*DB)*

Это нужно для хранения данных на сервере игры, что бы пользователь при возвращении в игру видел свои результаты после последнего входа, например, очки.

2 - Функционал логированния. (*Logs*)

Это нужно для того что бы во время работы сервера, можно было сохранять в журнал, что происходило, например, в 15:00 подключился игрок, в 15:05 он сделал ход. В дальнейшем это понадобиться для выяснения что происходило, например, если что-то сломается.

3 – Функционал соединение. (*Connector)*

Это нужно, что бы сервер и клиент могли обмениваться данными, общаться. Т.к. когда игрок нажимает какую-то кнопку, с помощью соединения отправляются программные сообщения на сервер, и сервер что-то с этим делает. Например, игрок нажал в поле и поставил крестик, после этого программное сообщение отправляется на сервер, сервер проверяет можно ли сделать ход в это место, и если да, то отправляет сопернику программное сообщение, далее у соперника появляется этот поставленный крестик.

4 – Загрузчик сервера, загрузчик клиента. (*Loader)*

Для сервера это будет программный код, который запустит весь остальной код, и заставит его работать. Так надо, это как ключ зажигания в машине…

Для клиента, это тоже самое почти. Когда мы открываем игру вКонтакте, сначала мы открываем именно загрузчик, он уже загружает все картинки, звуки, логику, кнопки и прочие.

5 – Роутер. (ApiRouter)

Тут мы будем маршрутизировать сообщения, какие связаны с игрой, какие и пользователями, а какие с чатом.

Этап 2. **Функционал для работы с игроками.**

**Результат:** Внешне результата опять никакого,☺ но это тоже нужно для дальнейшей разработки. Без этого мы не сможем отобразить сколько очков, сколько онлайн пользователей, да и вообще ничего не сможем, т.к. играть будет с точки зрения программного кода некому. ☺

1 – Таблица пользователей. (*DB.users*)

Это нужно для хранения на сервере данных о пользователях, их очках, например, онлайн или офлайн они и т.д.

2 – Авторизация через ВК. (*LogicUser.AuthorizationByVK*)

В каждой социальной сети свой собственный способ сообщить нашей игре, данные о пользователях, собственно это мы и будем делать, спрашивать у социальной сети, что за игрок зашел, какие-то его данные и сохранять это у нас на сервере, либо просто проверять, что такой игрок уже есть на сервере и знать, что это именно он зашел в игру.

3 – Выход игрока из игры. (*LogicUser.onDisconnect*)

Нам нужно сделать, что бы мы знали, что он вышел, и далее сообщать об этом другим пользователям, что бы они видели, сколько пользователей онлайн и не вышел ли вообще игрок во время игры, иначе игрок так и будет сидеть, и ждать хода оппонента.

4 – Информация об игроке. (*LogicUser.getInfo*)

Сюда входит вся возможная информация об игроке. Нам надо запрашивать у социальной сети имя игрока, его фотографию, возраст, пол и прочие, хранить это на сервере и рассылать другим игрокам эти данные, ну что бы они знали с кем они играют. Так ведь интересней ☺

5 – Запрос списка друзей. (*LogicUser.getFriendsList*)

Тут мы будем запрашивать у социальной сети списка друзей и хранить его у себя в игре. Эта нужно в т.ч. для того что бы друзья могли играть друг с другом – так интересней.

6 – Обновление онлайн\офлайн. (*LogicUser.updateOnline)*

Тут мы просто сделаем так, чтобы знать кто сейчас онлайн, а кто офлайн и рассылать эту информацию всем игрокам, пусть радуются, что они не одни сидят в этой игре ☺

Этап 3. **Функционал графики, базовый.**

**Результат:** Тут мы наконец то сможем увидеть немного игру, кнопку «играть», кол-во онлайн игроков, флажок выбора игры 3х3 или 15х15, флажок игры с роботом или нет, ну и пока что всё. ☺

1 – Контроллер страниц. (*PageController*)

Когда мы загрузим игру, мы увидим главную страницу, на ней будет кнопка «Играть», когда мы нажмем «играть» то переключимся на страницу с игрой, таким образом нам надо сделать так, чтобы в игре, на клиенте, у нас были разные страницы, ну и переключаться между ними конечно надо будет. Это мы и будем делать в этой задаче.

2 – Элемент Картинка. (*ElementImage*)

После выполнения этой задачи мы сможем выводить картинки на клиенте. Это и задний фон стола, и игровое поле и прочие, всё что может быть картинкой, даже кнопки будут использовать элемент картинки.

3 – Элемент Кнопка. (*ElementButton*)

Ну тут всё понятно, нам нужны кнопки, но для этого мы сначала сделаем возможность их создавать.

4 – Элемент Текст. (*ElementText*)

Нам надо будет отображать текст, это будет «очки: 123», «онлайн пользователей: 123» и т.д.

5 – Элемент Флаг. (*ElementCheckbox*)

Это галочка. Что бы указывать играем мы с роботом или без.

6 – Элемент Переключатель (англ. Radio button). (*ElementMultiCheckbox*)

Это такой графический элемент, когда мы можем выбрать что-то одно: т.е. когда на сайте у нас стоят варианты «Выберите один вариант» и когда мы выбираем другой вариант, то предыдущий отменяется, видимо по русский это называется «переключатель».

7 – Основная страница. (*MainPage*)

Тут мы создадим страницу, сделаем на ней задний фон, поместим на неё элементы: кнопку «Играть», текст кол-во онлайн пользователей и текст с очками пользователя. И пока что всё.

8 – Страница Игры. (*GamePage*)

Тут мы сделаем страницу игры, на неё будет игровое поле, кол-во онлайн пользователей, очки, сообщение о том, кто сейчас ходит\либо, кто победил\проиграл. Хотя пока что эту страницу видно не будет, он она будет существовать. ☺

Этап 4. **Функционал случайной игры с соперником.**

**Результат:** можно будет нажать кнопку «играть» на главной странице и перейти на страницу игры, там будет игровое поле. Можно будет открыв два браузера поиграть на поле 3х3 или 15х15. Можно будет выбирать каким знаком играть: крестиком, ноликом или случайным.

1 – Таблица игр. (*DB.games*)

Нужно, чтобы хранить игры, это понадобиться для рейтинга в будущем, для статистики и еще для чего-нибудь, в любом случае нельзя забывать об играх. Как-то это не кошерно, может в будущем мы захотим отобразить в игре историю игр.

2 – Функционал создания игры. (*LogicGame.createGame*)

Сделаем: При нажатии на кнопку «Играть» переходим на страницу игры, игра создается на сервере. Либо мы попадаем в уже созданную игру кем другим.

3 – Покинуть игру. (*LogicGame.closeGame*)

Сделаем: если игрок вышел, будем завершать игру и сообщать об этом сопернику. Соответственно игра будет уходить в статус: закрыта. Кстати возможные статусы: «ждём оппонента», «идёт игра», «ничья», «есть победитель», «закрыта»

4 – Ход игроком. (*LogicGame.doMove*)

Сделаем: можно будет делать ходы, определять кто сейчас ходит и отображать это в игре: «ваш ход», «ход оппонента».

5 – Проверка победителя. (*LogicGame.checkWinner*)

Сделаем: после каждого хода будет выполнятся проверка, есть ли победитель, соответственно игра будет переходит в один из статусов: «есть победитель», «ничья», либо оставаться в статусе «идёт игра».

6 – Начисление очков. (*LogicUser.onWin*)

Сделаем так что будут прибавляется очки при победе, а данные об этом будут рассылаться другим игрокам.

Этап 5. **Игра с роботом (ботом в простонародье).**

**Результат:** Можно будет поиграть с роботом, указав галку «С роботом».

1 – Создание игры с роботом. (*LogicRobot.createGame*)

Теперь при создании игры можно будет указать что это игра с роботом. Робот будет ходить, если его ход.

2 – Ход роботом. (*LogicRobot.AIDoMove#1*)

Тут мы научим робота делать ход в игре.

3 – Мозг робота. (*LogicRobot.AIDoMove#2*)

Это будет программный код, который будет уметь играть, не просто ставить крестики и нолики куда попало, но стремиться к выигрышу, и защите от проигрыша (мешать оппоненту). Обычно это называется искусственный интеллект.

4 – Проверка победителя. (*LogicRobot.checkWinner*)

Для игры с роботом нужна своя проверка. Она похожа на проверку в обычной игре (игрок с игроком), но всё же отличается: не оповещается оппонент, роботу, как оппоненту, не начисляются очки.

5 – Покинуть и игру с роботом. (*LogicRobot.closeGame*)

Функционал похож на выход из обычной игры. Но, всё же надо будет его доработать для робота. Доработка заключается лишь в том, что при выходи из игры мы не оповещаем оппонента. Роботу это не надо, как никак. ☺

Этап 6**. Функционал списка игр.**

**Результат:** Можно будет видеть список игроков, которые ждут игры и вступать с ними в игру. Выбрать конкретного игрока может быть интересно.

1 – Элемент списка игр. (*ElementGamesList*)

Графический элемент отображения списка игр. Это только именно графика: будем отображать: имя игрока, тип поля:3х3 или 15х15, каким знаком предлагается играть: крестиком или ноликом.

2 – Рассылка созданных игр(*LogicGame.sendAllGameInfo*)

При создании игроком игры, он будет ждать оппонента, сервер будет рассылать об этом информацию, так что все остальные игроки смогут видеть в списке игр, созданную игру и вступать с ним в игру.

3 – Вступление в игру из списка. (*LogicGame.joinGame*)

При нажатии на игру в списке, игрок будет вступать в игру, и всем будет рассылаться информация о том, что игра более не ждёт оппонента.

Этап 7. **Функционал чата.**

**Результат:** В игре появиться окно чата, можно будет писать сообщения. Круто!

1 – Элемент окно чата. (*ElementChatWindow*)

Графический элемент, в нём будут отобржаться сообщения. Для каждого сообщения будут следующие данные: Имя отправителя, время отправки, само сообщение.

2 – Элемент для ввода сообщения. (*ElementChatInput*)

Графический элемент, в него можно будет ввести текст, нажать кнопку Enter и отправить сообщение.

3 – Функционал отправки сообщений на сервер. (*LogicChat.sendMessage*)

Именно код, с помощью которого клиент сможет передавать программное-сообщение содержащие сообщение-чата (кто отправил и само сообщение).

4 – Рассылка сообщений игрокам. (*LogicChat.sendAllMessage*)

Код, который будет рассылать всем игрокам отправленное кем-то сообщение.

5 – Запрос последних сообщений(*LogicChat.getLastMessages*)

Когда пользователь зайдет в игру, надо будет ему отобразить последние 5-10 сообщений. Да так будет круче, пожалуй.

Этап 8. **Функционал отображения друзей.**

**Результат:** В игре можно будет увидеть список друзей, обычно это называется лента друзей. В списке будет фотография, онлайн-статус пользователя, занят (в игре) он или нет, очки пользователя. Пока всё. Так же мы сделаем фотку соперника во время игры с ним.

1 – Элемент Фото Игрока. (*ElementPlayerPhoto*)

Графический элемент, фотография, с возможность смотреть онлайн-статус, занят игрок или нет, его очки и имя.

2 – Элемент Лента друзей. (*ElementFriendsType*)

Графически элемент, который будет содержать в себе фотографии игроков, а именно друзей. Пять штук слева на права.

3 – Функционал загрузки в ленту-друзей данные о друзьях. (*LogicFriends.updateFriendType*)

Ну собственно весь код, который нужен, что бы в ленте отображались друзья и их актуальная информация.

Этап 9. **Функционал приглашений в игру.**

**Результат:** На ленте друзей появится кнопка пригласить, если нас пригласили, то на ленте друзей появиться текст: «играем»? при нажатии на этот текст оба игрока будут попадать в игру.

1 – На элемент Фото игрока добавим иконку пригласить, и мини кнопку «играем?». (*elementPlayerPhoto#2*)

Всё просто.

2 – Функционал отправки приглашения. (*LogicInvites.send*)

При нажатии на «Пригласить» будет отправлять на сервер запрос, а сервер будет отправлять игроку и у того будет отожраться надпись: «Играем?» под фото пригласившего.

3 – Функционал принятия приглашения. (*LogicInvites.accept*)

При нажатии «Играем?» будет создаваться игра, в которую войдут оба игрока.

4 – Функционал состояния пользователя. (APIUserState)

Так мы сможем сообщать всем игрокам, что пользователь занят, и в какой он игре находиться, дело в том, что без этого мы не сможем нормально приглашать пользователей в игру. Ведь, если он занят (в игре, например, находиться). То, приглашать его, просто нет смысла.

Этап 10. **Рейтинг.**

**Результат:** На клиенте, на главной странице появиться кнопка: «Рейтинг», нажав на неё мы попадём на страницу рейтинга, там будет список: Место, Имя, Фото, очки.

1 – Элемент список рейтинга. (*elementRatingList*)

Графический элемент на клиенте, список, в каждой строке будет: место в рейтинге, фото, имя и фамилия игрока, очки.

2 – таблица рейтинга. (*DB.rating*)

Таблица на сервере, где будет храниться рейтинг. Считать его каждый раз заново – большая нагрузка, сервер не справиться☺. Скажем, если 10 игроков откроют страницу рейтинга, всё зависнет скорей всего.

3 – Обновление рейтинга на клиенте. (*LogicRating.updateRating*)

Запрос у сервера списка рейтинга, и загрузка его в элемент списка рейтинга.

4 – Расчёт рейтинга. (*LogicRating.calculateRating*)

Сервер будет считать рейтинг, скорей всего каждый раз, когда кто-то выиграл. Или еще как иначе, пока сложно сказать, но это отдельная задача.

Этап 11. **Подсказки.**

**Результат:** На главной странице появиться кнопка знака вопроса в кружочке, нажав на неё мы попадём на страницу помощи, а ней будет рассказано как играть. Да, не все игроки так умны, примерно 85% нужна не просто страница помощи, а даже целый визард! **Визард** – по-русски это помощник-обучающий, т.е. если заходишь в игру и появляется всякие: кликни сюда, молодец! а теперь сюда, молодец получи бонус! Это и есть визард (англ. Wizard).

1 – Иконка помощи на главной странице. (*imageHelp*)

Тут просто поместим на главную страницу иконку, добавим страницу помощи, и возможность переходить на страницу помощи, а также возвращаться на главную страницу.

2 – Страница помощи. (*PageHelp*)

Собственно, сама помощь, это текст и картинки.

Этап 12. **Тестирование и выкладка.**

**Результат**: мы сможем убедиться, что игра выдержит нагрузку большого числа игроков. А также выложим игру в контакт, но еще не будем отправлять на одобрение. **Одобрение вКонтакте –** игру можно выложить сразу, но в неё нельзя будет приглашать друзей, это большой минус, и игра не будет в общем каталоге, откуда собственно приходят пользователи. Игру можно отправить на одобрение, тогда модератор вКонтакте проверит в игру, поиграет в неё и если даст одобрение. То игра появиться в общем каталоге, что вообще очень важно, иначе откуда взяться участникам.

1 – Выкладка игры во вКонтакте.

Собственно, выложим игру вКонтакте, так что её можно зайти и потыкать-таки. Однако пока что это можно будет сделать только с моего компьютера. Позже будем переносить на сервер в интернете, так что зайти можно будет, когда угодно и откуда угодно, где только есть интернет.

2 – Статистика.

Сделаем компонент Статистики, чтобы следить за посещаемостью, и какие именно места интересны пользователям, например, мы сможем узнать, что из 10000 человек на поле 3х3 играет всего 1, тогда мы уберем это поле. Т.к. оно бессмысленно по сути будет.

2.1. – Таблица статистики.

Что бы хранить данные на сервере.

2.2. – Компонент статистики.

Код, с помощью которого можно писать собственно саму статистику.

2.3. – Выставление статистик.

На разные действия будем писать статистику. Например, когда пользователь вошел в игру, в статистике будет отображаться: Игрок такой то, вошел в игру во столько-то. И когда он вышел из игры. Так мы сможем узнавать, например, как долго пользователи держаться в игре и т.д.

3 – Ручное тестирование.

Проверим руками, всё что в голову придёт, это необходимая процедура, в этом процессе мы точно выявим баги и будем их править. Так уж сделан мир, ахах ☺

4 – Авто тестирование.

Создадим авто тесты сервера, они будут эмулировать запросы к серверу от клиента. Это нужно для гарантии, что ничего не отвалилось, во-первых, мы проведем само тестирование, и скорей всего найдем пару багов. Во-вторых, в будущем, когда мы будем менять что-то в коде. Сможем запустить эти тесты, и они проверят, что ничего не поломалось. Так устроено программирование, что когда меняешь в одной части кода, то может сломаться в другом месте, и обычно об этом не узнают до тех пор, пока игрок не сообщит об этом, а он конечно же будет молчать! Ну а что бы мы знали о том, что что-то сломалось, у нас будут авто тесты.

5 – Тестирование нагрузки.

Сделаем Код, который будет эмулировать действия игроков, и затем зададим сколько именно игроков будут «играть». Будем проверять, сначала симулируем 1-ого пользователя, потом 10 пользователей, потом 50 пользователей, 100 и 500 и т.д. Пока не достигнем того, что сервер не справиться с нагрузкой и зависнет, тогда мы узнаем, при помощи логов (задача в 1 этапе - логирование) почему сервер завис и скорей всего поправим как-то это, что бы сервер мог нести максимально большую нагрузку.

6 – Перенос на хостинг.

Нам понадобиться где-то рублей 128-2048 в месяц, на то что бы поместить игру в интернете, и на неё можно было заходить откуда угодно и когда угодно (24ч в сутки). Ну деньги тут копейки для такой радости. Далее конечно надо будет все проверить, так же ли оно работает, как и на моём компьютере. Для этого мы заново запустим тесты.

**Продолжение следует…**

- Дело в том, что в процессе всей разработки, у нас будут появляется новые идеи, это неизбежно.

- Вообще надо сказать, что некоторые этапы зависят друг от друга, т.е. нельзя выполнить этап чат без этапа базового функционала графики и т.д., однако некоторые этапы можно менять местами.

- После завершения каждого этапа. Мы будем выбирать какой этап выполнять следующем, если конечно он не зависит от каких-то еще не сделанных этапов.

- Однако, часто делают так, что, придумав новую идею, начинают её делать сразу, это ошибка.

- На практике, эти новые идеи забивают время до бесконечности, и мы просто не сдвинемся с места, если будет делать все что приходит в голову, но не входит в какой-то этап.

- Так что, все новые идеи мы будем записывать и откладывать до конца завершения всех этапов текущего плана. Далее мы, наверное, будем уже делать просто задачи, но будем тоже их группировать как-то.

- Так же, когда мы выполним все этапы и посмотрим на то что, получилось, будем думать, что доделать или изменить, а нам обязательно захочется это сделать.

- Функционал на будущие:

- замороженная игра;

- история;

- Рейтинг от моей позиции, рейтинг топ, рейтинг среди друзей;

- Постинг на стену результатов игры.

- Приглашение друзей, кстати забыл это включить, по любому продеться сделать.

- Выкладка в другие соц сети. Одноклассники, фейсбук, LinkEdin и прочие.

- Реализация для смартфонов;

- реализация для планшетов;

- убирания чата;

- полноэкранный режим;

- доработка для игры с сенсорным экраном;

- переделывание графики шрифта;

- переделывание графики заднего фона игры;

- выкладка на гугл плей или аппс, или как его там, ну суть понятна;

- регистрация в рейтинге игр, есть сайт, не помню как называется, там можно узнать статистику игр, это нужно для мониторинга, что происходит с нашей игрой, и дальнейшего прогнозирования и т.д. А так же например, для будущего резюме.

- создания группы для игры;

- рекламирование игры через вКонтакт;

- рекламирование через группы, узнать стоит ли вообще это делать;

**Содержание:**

Описание документа.

Этап 1. Базовый функционал.

Этап 2. Функционал для работы с игроками.

Этап 3. Функционал графики, базовый.

Этап 4. Функционал случайной игры с соперником.

Этап 5. Игра с роботом (ботом в простонародье).

Этап 6. Функционал списка игр.

Этап 7. Функционал чата.

Этап 8. Функционал отображения друзей.

Этап 9. Функционал приглашений в игру.

Этап 10. Рейтинг.

Этап 11. Подсказки.

Этап 12. Тестирование и выкладка.

Содержание.