УДК 004.75

В.А. Алексеева (5 курс, каф. ПМИ), В.И. Грищенко, доцент каф.ПМИ.

СОВРЕМЕННЫЕ ПОДХОДЫ К ПОСТРОЕНИЮ СЕТЕВЫХ ИГРОВЫХ ПРИЛОЖЕНИЙ

В рамках дипломной работы будет разрабатываться сетевая онлайн игра дляоперационной системыAndroid.Сетевое (распределенное) приложение, представляет собой несколько взаимосвязанных частей, каждая из которых выполняет определенную задачу, которая может выполняться на отдельном компьютере сети. Для взаимодействия частей сетевого приложения используются сетевые службы и транспортные средства ОС.

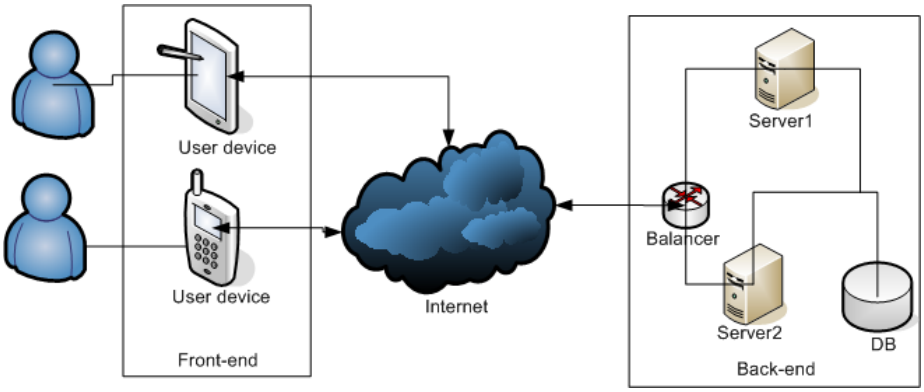

Рис. 1 Упрощенная схема сетевого приложения

Пользователь запускает клиентскую часть системы на смартфоне или планшете. Клиентская часть выполнена в виде приложения на языке Java. Приложение в ходе своей работы обращается к серверной части системы (Back-end) для актуализации игровой сцены и сохранения промежуточных результатов. Back-end состоит из балансировщика запросов, кластера серверов приложений и СУБД.

При разработке серверной части приложения необходимо уделить внимание ряду важных задач: балансировка нагрузки на узлы системы, обеспечение пользователя всегда точной, достоверной и актуальной информацией, восстановление данных в случае возникновения ошибки, обеспечение быстродействия и масштабируемости системы. Согласно с требованиями к back-end на рисунке 1 приведена схема, которая отображает общую структуру приложения с точки зрения обработки информации.

Прежде чем начать разработку приложения необходимо определить, программные средства (язык программирования, тип базы данных, средства для передачи данных между front-endи back-end), которые будут использоваться для разработки приложения.

1. Выбор базы данных для системы

Базы данных, на данный момент, используются во многих приложениях, начиная с самых простых и заканчивая бизнес-приложениями. В современной индустрии получили распространение два типа БД: реляционные и NoSQL.

К реляционным базам данных относятся (MySQL, SQLite и PostgreSQL), выделим их основные характеристики:

– реляционная база данных является совокупностью взаимосвязанных таблиц;
– каждая таблица содержит данные об объектах определенного типа;
– столбцы таблицы описывают характеристики объекта;
– одинаковая структура записей в рамках одной таблицы;

– все данные строго типизированы;

Реляционные БД обеспечивают:

– атомарность (результаты всех операций либо отображаются в состоянии базы, если успешны, либо нет, если операция не была успешной);

– согласованность (успешное завершение транзакции с фиксацией результатов только в том случае, если действия операции не нарушат целостность БД);

– изолированность (обеспечивает независимость транзакций при их параллельном выполнении, т.е. транзакции никак не воздействуют друг на друга, пока не будут успешно завершены);

– надежность (все изменения внесенные операциями в рамках одной транзакции гарантированно сохраняются) [1].

NoSQL базы данных делятся на:

– документно-ориентированные БД (представляют собой по структуре дерево, в листах содержатся данные, которые при добавлении документа заносятся в индексы. Выборка в таких БД позволяет получить части большого количества документов без их полной загрузки) MongoDB, CouchDB;

– БД на основе графов (база данных, которая хранит данные в виде графа и элементами являются узлы и связи. Представляет собой структуру, где взаимосвязанные данные соединены без применения индекса) OrientDB, Neo4J;

– хранилище ключ-значение (работает с данными типа ключ-значение. При обращении к серверу приложение задает ключ и соответствующее значение, а после получает данные по ключу. Удобны при хранении сессий, кеша и т.д.) Redis, MemcacheDB[2];

Данные в таких БД представляются в виде агрегатов (вместо использования логической последовательности, все объекты рассматриваются как единое целое). Они легко подвергаются горизонтальному масштабированию и используют специализированный механизм запросов вместо SQL.

Таблица 1 – Достоинства и недостатки БД

	Преимущества	Недостатки
NoSQL базы данных	– быстрое считывание информации; – хранение объектов в удобном виде для приложения (агрегаты вместо таблиц); – поддержка атомарности на уровне записей;	– использование только определенного вида запроса; – сложности, в случае, если один и тот же объект может выполнять различные роли;
Реляционные базы данных	– сохраняется целостность информации при обновлении; – возможность работы с различными запросами;	– медленное чтение при использовании соединений таблиц; – сложность горизонтального масштабирования;

В результате, для разрабатываемого приложения была выбрана реляционная БД, т.к.поддержание необходимых связей между данными в таблицах вызывает сложности при использовании NoSQL. Например, если необходимо вывести список игроков, с которыми пользователь сегодня играл, то будет нужно продублировать информацию в файлах обоих игроков, а в реляционной БД это лишь одна запись в таблице пересечения инет необходимости отслеживать сложные зависимости между игроками, поэтому БД на основе графов тоже не подходят.

2. Выбор языка программирования для сервера

Наиболее популярными языками программирования для распределенных приложений являются Python, Ruby и PHP.

Python применяется как интерпретируемый язык для скриптов различного назначения. Основной целью Python является приближение синтаксиса реальной программы, написанной на нём, к описывающему задачу псевдокоду, что позволяет уменьшить объём программы. Python позволяет совместить два подхода к программированию: процедурный и объектно-ориентированный.

Ruby – динамический императивный объектно-ориентированный язык программирования, разработанный Юкихиро Матсумото.

PHP – язык программирования, исполняемый на стороне веб-сервера, спроектированный Расмусом Лердорфом (Rasmus Lerdorf) в качестве инструмента создания динамических и интерактивных веб-сайтов. Является мощным и гибким из-за чего и стал одним из популярнейших языков. Возможности языка позволяют применять его в проектах любого масштаба[3].

Для выбора из этих языков наиболее подходящего, их достоинства и недостатки были представлены в виде таблицы 2.

В результате, для системы был выбран PHP, так как он легок в освоении, является кроссплатформенным и очень популярен при разработке веб-приложений.

Таблица 2 – Преимущества и недостатки Python, Ruby, PHP

	Преимущества	Недостатки
Python	– открытая разработка; –прост в изучении; –стимулирует писать читаемый код; – большое сообщество, которое позитивно к новичкам; – множество полезных библиотек и расширений.	– мало качественных программ; –малая производительность в сравнении с другими языками.
Ruby	–открытая разработка; –работает на многих платформах; –реализует концептуально чистую объектно-ориентированную парадигму; –предоставляет продвинутые методы манипуляции строками и текстом; –возможности языка можно расширить при помощи библиотек.	–сложности в изучении языка для новичков; –малая производительность в сравнении с другими языками; –медленно развивается.
PHP	–является свободным программным обеспечением, распространяемым под особой лицензией (PHP license); – легок в освоении; – большая аудитория пользователей; – обширный набор средств, при работе с базами данных; – множество дополнительных библиотек и настроек; –может быть развёрнут почти на любом сервере; –поддерживается на многих платформах и операционных системах.	–не подходит для создания десктопных приложений или системных компонентов; –глобальные параметры конфигурации влияют на базовый синтаксис языка, что затрудняет настройку сервера и разворачивание приложений;

3. Выбор метода взаимодействия компонентов приложения

Наиболее популярными методами являются: REST(метод взаимодействия, при котором вызов удаленной процедуры представляет собой HTTP-запрос, а данные являются параметрами запроса) и SOAP (протокол, основанный на XML и предназначен для обмена структурированной информацией между распределенными приложениями поверх существующих веб протоколов). Исходя из данных приведенных в таблице 3 и задач, которые необходимо решить наиболее подходящим будет метод взаимодействиякомпонентов через REST [4].

Таблица 3 – Преимущества и недостатки SOAPи REST

	Преимущества	Недостатки
REST	– производительность; – простота использования; – простота разработки и добавления функциональности приложению; – надежность (нет необходимости сохранять информацию о состоянии клиента); – малая ресурсоемкость; – ответ может быть представлен в разных формах.	– не стандартизирован
SOAP	– при передаче данных использует XML, что позволяет пользоваться всеми возможностями, предоставляемыми этим языком; – может использоваться как поверх других протоколов.	– увеличение объема сообщения, что снижает скорость передачи.

Выводы

Данное исследование является частью дипломного проекта и позволяет получить представление о задачах, которые возникают перед разработчиком распределенного приложения. В соответствии с ними были проанализированы и выбраны подходящие средства разработки и составлена полная схема распределенного приложения.

ЛИТЕРАТУРА:

1. Базы данных. Вводный курс [Электронный ресурс]. – Режим доступа: http://citforum.ck.ua/database/advanced_intro/39.shtml

2. Сравнение NoSQL систем управления базами данных [Электронный ресурс]. – Режим доступа: http://devacademy.ru/posts/nosql/

3. PHP, Ruby, Python краткая характеристика [Электронный ресурс]. – Режим доступа: http://www.internet-technologies.ru/articles/article_1991.html

4. RESTvsSOAP [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/131343/