ПРОГРАММНЫЙ КОМПЛЕКС «REMEMBER ME» В СРЕДЕ ОС ANDROID

Кошеутова Н.В., Осина П.М.
Научный руководитель: Шерстнев В.С., к.т.н., доцент кафедры ВТ
Томский политехнический университет
E-mail: polinaosina14@gmail.com, nat.dar@mail.ru

Введение

Управление временем для современного человека является основным понятием, благодаря которому достигается эффективность и продуктивность любых процессов. Тот, кто добился успеха в своей жизни, много времени посвящают планированию. Ежедневное планирование просто необходимо для повышения производительности и эффективного управления временем.

В современном мире каждый первый человек имеет смартфон, на котором могут быть установлены приложения–органайзеры. Благодаря данным приложениям планирование дел становиться удобным, быстрым, а главное мобильным. Но, к сожалению не все приложения обладают полным функционалом, который пригодился всем, включая домохозяек, бизнесменов, преподавателей и студентов.

Описание архитектуры

Архитектура программного комплекса должна быть гибкой и обеспечивать простое и быстрое взаимодействие клиентов с сервером, а также с сервисами GoogleMap и сайтом расписаний для экспорта расписания.

Архитектура изображена на рисунке 1. Для осуществления возможности использовать данный комплекс на любом устройстве необходимо разработать два клиентских приложения для пользователей: мобильное приложение для Android OS и web-приложение для браузера. В реализации данного вида архитектуры информация всех пользователей хранится в базе данных на сервере. Информация пользователей мобильного устройства хранится в локальной базе данных на устройстве. Web-сервис осуществляет связь между клиентами и базой данных севера, предоставляя готовую информацию. Также web-сервис осуществляет экспорт расписания с сайта ТПУ в базу данных на сервере.

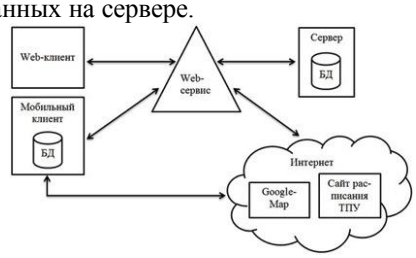

Рис. 1 Архитектура программного продукта

К достоинствам данной архитектуры можно отнести: независимость от используемых клиентских платформ, независимость от применяемых языков программирования, наличие с единообразного доступа к web-сервису для всех клиентов. Также достоинством является масштабируемость системы, добавление новых модулей функций, вынесение логики взаимодействия на web-сервис.

К недостаткам данной архитектуры можно отнести громоздкую структуру.

Инструменты разработки программного продукта

На основе разработанной архитектуры и основных функций будущего программного комплекса были выбраны следующие средства разработки:

1. Для разработки мобильного клиента: Android Studio, язык программирования Java;
2. Для разработки базы данных: для web-сервера – MySql, для мобильного приложения – SQLite;
3. Для разработки web-сервис – язык программирования PHP;
4. Для разработки web-клиент – язык программирования PHP, язык разметки гипертекста HTML.

Разработка БД

После определения архитектуры разрабатываемого приложения необходимо разработать концептуальную, логическую и физическую модель БД, основываясь на сфере применения БД и систем для которых БД будет разработана.

Для разработки БД для приложения под Android OS была использована SQLite. SQLite – это компактная встраиваемая реляционная БД.

Для разработки БД в web-приложении и сервере была использована MySQL. Так как для разработки сервера и web-приложения используется пакет программ WAMP (Windows, Apache, MySQL и PHP).

Определившись с СУБД можно перейти к разработке самой БД. Разработка концептуальной модели БД должна соответствовать некоторым правилам:

1. Определиться с информацией, хранимой и обрабатываемой в БД;
2. Преобразовать данную информацию в сущность;
3. Дать имя сущности;
4. Разбить сущность на атрибуты;
5. Определить уникальный идентификатор сущности;
6. Далее необходимо определить связи между экземплярами сущности;

Так же необходимо чтобы БД соответствовала правилам нормализации. Нормализация БД заключается в процессе организации данных, включающий создании таблиц и установлении отношений между ними таким образом, чтобы избежать избыточность, несогласованные зависимости и обеспечить гибкость БД.

После разработки концептуально модели можно перейти к разработке физической модели (Рис. 2).

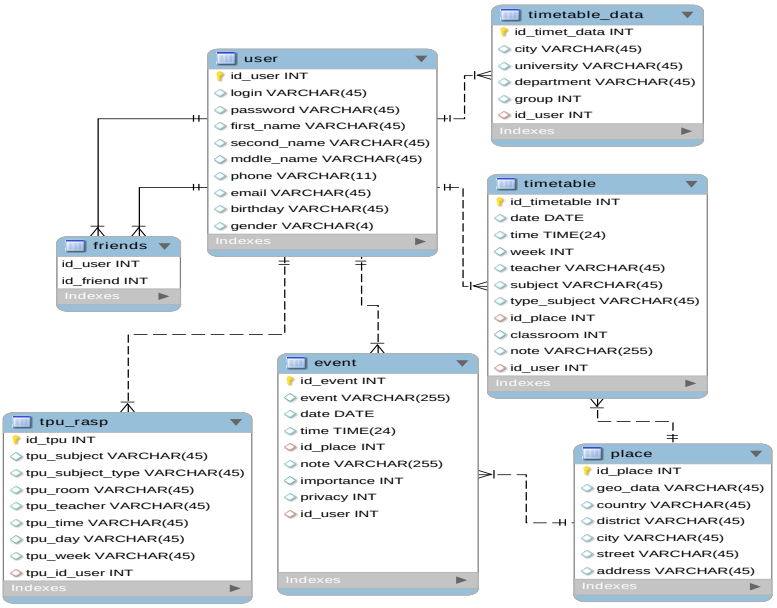

Рис.2. БД

Разработка web-сервиса

В архитектуре данного программного комплекса связующим звеном между клиентскими частями и базой данных является Web-сервис.

Web-сервис является общим термином для международных стандартов, которые сформированы на базе языка XML и позволяют конечным узлам обмениваться данными и выполнять определённые функции, операции через Интернет. Основные функции Web-сервисов позволяют создавать информационные услуги доступа к серверам или другим компьютерам. Формат сообщения на базе языка XML используется для «упаковывания» запросов и данных в целях взаимопонимания между обоими участниками общения.

Web-сервис обеспечивает взаимодействие программных систем независимо от платформы. Например, в данном случае AndroidOS-клиент может взаимодействовать с PHP-сервером, работающим под Windows. Так как web-сервисы основаны на базе открытых стандартов и протоколов. Благодаря использованию XML достигается простота разработки и отладки web-сервисов.

Основными компонентами веб-сервисов являются:

- сервер приложений, где размещается сервис (то есть где функционирует ПО сервера);
- интерфейс сервиса (часто описываемый на языке веб-сервисов);
- хранилище данных или справочник с описанием интерфейса на WSDL, чтобы клиенты веб-сервисов могли найти (и использовать) интерфейс;
- клиент веб-сервиса, желающий использовать веб-сервис;
- протокол связи, позволяющий клиенту веб-сервиса общаться с сервисом.

Существует два пути реализации Web-сервиса данного программного комплекса, первый путь – с использованием протокола SOAP, второй – с использованием архитектурного стиля взаимодействия компонентов REST.

SOAP – это целое семейство протоколов и стандартов. На данный момент SOAP используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур. SOAP является расширением протокола XML-RPC. SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS . Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно. Чаще всего SOAP используется поверх HTTP.

REST — это не стандарт и не спецификация, а архитектурный стиль, выстроенный на существующих, хорошо известных и контролируемых консорциумом W3C стандартах, таких, как HTTP, URI (Uniform Resource Identifier), XML и RDF (Resource Description Format). В REST-сервисах акцент сделан на доступ к ресурсам, а не

на исполнение удаленных сервисов; REST целиком и полностью основан на протоколе передачи данных. Наиболее распространенный протокол HTTP. Для HTTP протокола действие над данными задается с помощью методов: GET (получить), PUT (добавить, заменить), POST (добавить, изменить, удалить), DELETE (удалить). Таким образом, действия CRUD (Create-Read-Updtae-Delete) могут выполняться как со всеми 4-мя методами, так и только с помощью GET и POST.

Отличия SOAP и REST безусловно отличаются друг от друга, если SOAP-клиенты запрашивают выполнение действия на сервере, то REST-клиенты попросту требуют сам ресурс.

SOAP активно использует XML для кодирования запросов и ответов, а также строгую типизацию данных, гарантирующую их целостность при передаче между клиентом и сервером. С другой стороны, запросы и ответы в REST могут передаваться в ASCII, XML, JSON или любых других форматах, распознаваемых одновременно и клиентом, и сервером. Кроме того, в модели REST отсутствуют встроенные требования к типизации данных. В результате пакеты запросов и ответов в REST имеют намного меньшие размеры, чем соответствующие им пакеты SOAP.

В данной разработке выбран второй путь реализации web-сервиса с использованием архитектуры REST, так ка кона достаточно простота в реализации, пакеты запросов и ответов в REST имеют меньшие размеры.

Защита данных

После выбора средств разработки стал вопрос обеспечения безопасности хранения и передачи данных. Для этого были исследованы основные методы, используемые в защите информации в целом, а также какие способы реализованы в выбранных нами средствах разработки. [1]

Самой главной статьей в обеспечении безопасности является хэширование паролей, данную операцию необходимо проводить при разработке приложений, которые принимают пароли от пользователей. Без хэширования пароли могут быть украдены из базы данных, и все пользователи останутся без своих профилей в системе. [2]

Многие разработчики хэшируют пароли пользователей с помощью популярных функций, таких как *md5()* и *sha1()*. Такие хэширующие алгоритмы как MD5, SHA1 и SHA256 очень быстрые и эффективные. Но при наличии современных технологий и оборудования, стало довольно просто выяснить результат этих алгоритмов. Из-за той скорости, с которой современные компьютеры могут "обратить" эти хэширующие алгоритмы, многие профессионалы компьютерной безопасности строго не рекомендуют использовать их для хэширования паролей [3].

Существует несколько способов наиболее надежного хэширования паролей: первый из них – использовать несколько раз функцию *md5()* или *sha1(),* например *md5(md5($password));* второй способ – совмещать две функции, например, *sha1(md5($pass));* третий – способ использовать функцию *crypt(),* которая поддерживает несколько алгоритмов хэширования в PHP 5.3 и новее. Функция *crypt()* имеет параметр *salt* – это кусочек дополнительных данных, которые делают хэши более устойчивыми к взлому.

Наиболее из распространенных уязвимостей баз данных являются SQL-инъекции. SQL-инъекция – это разновидность уязвимости, которая позволяет заменить sql-запрос инородными данными. Защититься от данной уязвимости можно несколькими способами, и самые простые из них: первый способ – не вставлять напрямую переменную, которую ввел пользователь в SQL-запрос, а пропустить ее через такие функции как, *mysql_real_escape_string*(), которая экранирует специальные символы в строке [4]; второй способ – использовать встроенные функции PHP, для подготовки SQL-запросов, например, *mysqli_prepare() п*одготавливает SQL запрос и возвращает указатель на это выражение, который может использоваться для дальнейших операций с этим выражением, в случае если запрос содержит ошибку, данная функция возвращает значение false [5].

Заключение

Подведя итог обзорно-аналитической части, были сформированы основные цели и задачи «Выпускной квалификационной работы». Так же была освещена актуальность и практическая значимость разработки. Представлена архитектура и предполагаемые функции программного комплекса.

Список литературы

1. Мао В. Современная криптография. Теория и практика. М.: Вильямс, 2005. 763 с.

2. Рэнди Джей Яргер, Джордж Риз, Тим Кинг, MySQL и mSQL. Базы данных для небольших предприятий и Интернета. Издано: 2000, СПб, Символ-Плюс, ISBN: 5-93286-010-3, 560 стр.

3. PHP, безопасное хэширование паролей // php.net // URL: http://php.net/manual/ru/faq.passwords.php/ // (Дата обращения: 10.03.2016).

4. PHP, SQL-инъекции // php.net // URL: http://php.net/manual/ru/security.database.sql-injection.php/ // (Дата обращения: 10.03.2016).

5. Руководство по PHP // php.net // URL: http://php.net/manual/ru/mysqli.prepare.php/ // (Дата обращения: 17.03.2016).