Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ИЗУЧЕНИЕ ВОЗМОЖНОСТЕЙ QT 5.10.0 ПРИ СОЗДАНИИ КРОСС-ПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКОВ QML И C++

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Харламова Никиты Анатольевича

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

ВВЕДЕНИЕ

Целью данной бакалаврской работы является изучение возможностей Qt 5.10.0 при создании кросс-платформенных приложений с использованием языков QML и C++ в среде QtCreator. Qt - это кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. QML (Qt Meta Language or Qt Modeling Language) — декларативный язык программирования, основанный на JavaScript, предназначенный для дизайна приложений, делающих основной упор на пользовательский интерфейс. QML является частью Qt Quick, среды разработки пользовательского интерфейса, распространяемой вместе с Qt. В основном используется для создания приложений, ориентированных на мобильные устройства с сенсорным управлением однако, благодаря своей гибкости, успешно применяется при разработке всех видов приложений. Поставленные задачи:

- изучить рабочую область и функциональные возможности фреймворка Qt 5.10.0;
- реализовать внутренний и внешний интерфейсы приложения;
- реализовать связь приложения с SQLite базой данных;
- реализовать инструментарий пользователя для работы с картами;
- интегрировать в приложение функционал для работы с мультимедийными файлами;

Структура работы:

- Введение;
- Теоретическая часть;
- Экспериментальная часть;
- Заключение;

Теоретическая и экспериментальная части состоят из нескольких параграфов, а именно:

- Qt;
- Используемые Qt-модули;
- QtQuick;
- QtCreator;
- Разработка внутреннего интерфейса;
- Внутреннее устройство приложения;

1 Qt

Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Система свойств Qt основывается на возможностях самоанализа объектов во время выполнения программы. Самоанализ означает способность перечислить методы и свойства объекта и иметь всю информацию про них, в частности, о типах их аргументов. На этом принципе основываются языки QtScript и QML. C++ не предоставляет поддержки самоанализа, поэтому Qt поставляется с инструментом, который это обеспечивает. Этот инструмент называется MOC и является кодогенератором. Он генерирует дополнительный C++ файл, который компилируется с остальной частью программы. Этот сгенерированный C++ файл содержит всю информацию, необходимую для самоанализа.

Родственность Qt-объектов осуществляется через наследование класса QObject. QObject - это базовый класс для всех объектов Qt, его наследует любой класс использующий сигналы и слоты. Он обеспечивает возможность соединения объектов друг с другом. А также предоставляет к этому полезную функциональность. Во введении этого базового класса находится:

- прослеживание за потомками и родителями и возвращение указателей на них (Наследственная информация);
- программная реализация таймера;
- динамические свойства;
- интернационализация приложения;

В программировании графического интерфейса, когда меняется один виджет, часто необходимо, чтобы другой виджет получил об этом уведомление. В общем случае, чтобы объекты любого типа могли общаться с другими. Обычно такого рода связь получается путем обратного вызова. Обратный вызов это указатель на функцию, таким образом, если мы хотим чтобы функция уведо-

мила нас о каких-нибудь событиях, мы передаем указатель на другую функцию (обратновызываемую) этой функции. Функция в таком случае делает обратный вызов при необходимости.

В Qt же используется другая техника — сигналы и слоты. Сигнал вырабатывается когда происходит определенное событие. Слот это функция, которая вызывается в ответ на определенный сигнал. Виджеты Qt имеют много предопределенных сигналов и слотов, однако, при необходимости, возможно добавить дополнительные слоты и сигналы для дочерних классов. Механизм сигналов и слотов типобезопасен. Сигнатура сигнала должна строго совпадать с сигнатурой слота-получателя. Так как сигнатуры сравнимы, компилятор может помочь нам обнаружить несовпадение типов. Сигналы и слоты слабо связаны. Класс, который вырабатывает сигнал не знает, какие слоты его получат. Но механизм сигналов и слотов Qt гарантирует, что если мы подключим сигнал к слоту, слот будет вызван с параметрами сигнала в нужное время.

Слоты могут быть использованы для получения сигналов, но они также нормальные функции-члены. Слот ничего не знает о сигналах, которые к нему подключены. Это гарантирует, что с помощью Qt могут быть созданы полностью независимые компоненты. К одному слоту может быть подключено неограниченное количество сигналов, а один сигнал может быть подключен к неограниченному числу слотов. Также существует возможность подключать сигналы друг к другу. Это приведет к вызову второго сигнала сразу за первым. Таким образом сигналы и слоты вместе составляют мощный механизм создания компонентов.

2 Используемые Qt-модули

Qt-модули представляют собой основу Qt на всех платформах. Каждый из них представляет собой отдельную подключаемую библиотеку. Доступные на всех поддерживаемых платформах разработки и на всех тестируемых целевых платформах, они являются совместимыми для любой версии Qt 5.

Основные Qt-модули являются одинаково полезными для большинства Qt-приложений. А те модули, что используются для узко-специализированных задач, называются дополнениями, несмотря на то, что они тоже являются доступными для всех поддерживаемых платформ разработки.

Поддержка мультимедийных файлов в Qt обеспечивается модулем Qt Multimedia. Qt Multimedia предоставляет доступ к большому количеству функций и обеспечивает легкую работу с видео и аудио устройствами.

Некоторые функции:

- Доступ к звуковым устройствам для записи и воспроизведения
- Воспроизведение звука
- Создание плейлистов
- Запись и обработка звука
- Подключение к радиостанциям

QtSQL - один из основных модулей, который предоставляет поддержку SQL баз данных в Qt. Классы в QtSQL делятся на три уровня:

1. Уровень драйверов. Содержит классы QSqlDriver, QSqlDriverPlugin, QSqlResult QSqlDriverCreator и QSqlDriverCreatorBase. Этот слой предоставляет низкоуровневую связь между определенными базами данных и слоем SQL API. Вместе с Qt поставляются драйвера для MySQL, IBM DB2 (версии 7.1 и выше), PostgreSQL (версии 7.3 и выше), SQLite 2, SQLite 3 и других.
2. Уровень SQL API. Эти классы предоставляют доступ к базам данных. Соединения устанавливаются с помощью класса QSqlDatabase.

   Создать соединение, т.е. по сути создать экземпляр класса QSqlDatabase, можно с помощью одной из статических функций addDatabase() при задании используемого драйвера или типа драйвера и имени соединения. Соединение известно и поддерживает обращение к нему под своим собственным именем, а не по имени базы данных с которой оно соединяет. Одновременно может существовать множество соединений с одной и

той же базой данных. Помимо этого, QSqlDatabase также поддерживает вид соединения по умолчанию, которое по сути является неименованным соединением. Для создания соединения по умолчанию при вызове addDatabase() достаточно не передавать аргумент названия соединения. Далее, при вызове любой статической функции члена, принимающей имя соединения в качестве аргумента, с пустым аргументом, будет использовано соединение по умолчанию.

Само взаимодействие с базой данных осуществляется с помощью класса QSqlQuery. Он предоставляет возможности для выполнения SQL-выражений и манипулирования ими. QSqlQuery инкапсулирует функционал, вовлеченный в создание, направление и обработку данных из SQL запросов, обращенных к QSqlDatabase. Он может быть использован как для исполнения DML-выражений, таких как SELECT, INSERT, UPDATE и DELETE, так и DDL-выражений, к примеру CREATE TABLE. Также он может быть использован для исполнения специальных команд баз данных (SET DATESTYLE=ISO в PostgreSQL).

Сочетание двух модулей-дополнений QtPositioning и QtLocation позволяет получить доступ к широчайшему функционалу для реализации работы с картами, положением и прочим. QtLocation напрямую зависит от дополнения QtPositioning, поэтому каждое QML приложение использующее QtLocation, должно также импортировать QtPositioning.

QtPositioning обеспечивает возможность определять местоположение через информацию, получаемую через перечень источников, включающий в себя спутники, WiFi, текстовые файлы и прочее.

QtLocation - дополнение, которое помогает разработчикам, при использовании данных из популярных картографических сервисов, создавать готовые решения по работе с картами. Среди его функций:

- получение доступа и презентация картографических данных;
- поддержка запросов к определенным местоположениям и маршрутам;
- добавление на карту дополнительных слоев;
- поиск по местам и соответствующим им картинкам;

Конечный результат работы этих двух подмодулей - набор объектов, содержащих в себе данные о долготе, широте, высоте, скорости движения,

дату, время и другую сопутствующую информацию.

3 QtQuick

Основной идеей технологии является представление интерфейса в виде визуального иерархического дерева элементов, а также предоставление богатого набора переходов и анимаций, которые существенно облегчают процесс разработки. В Qt Quick используется основанный на JavaScript язык QML, позволяющий определять иерархию объектов, их внешний вид и свойства. Отличительной особенностью технологии является предоставление множества способов обмена данными между C++ и QML, что и обеспечивает необходимое изменение внешнего вида элементов при изменении их свойств.

На данный момент существует две версии QtQuick: QtQuick 1.0 и QtQuick 2.0. Во второй версии были улучшены используемые библиотеки, добавлены новые элементы, а также появилась поддержка API OpenGL 2.0. Библиотека компонентов QtQuick предоставляет богатый набор анимационных элементов, элементов для обработки изображений, элементов обработки ввода и модель-представление элементов. Особое внимание уделяется специальным пользовательским интерфейсам, созданным в рамках встраиваемых и мобильных устройств. Есть основанные на якорях макеты и позиционеры, но есть и компоненты высокого уровня, такие как поля со списком или панели инструментов.

QML это декларативный язык, который определяет иерархию объектов. Дерево объектов создается при загрузке QML файла. Для каждого объекта определяется набор свойств. В качестве значений свойств используются константы или выражения JavaScript. Значение свойств установленных как javascript вырження переоцениваются, если любые свойства используемые в выражении изменяются. Эта концепция декларирования свойств, которые зависят друг от друга и обновляются в реальном времени называется связыванием свойств (property binding) в QML. QML называют расширением JavaScript, но это также и улучшение. Хотя язык JavaScript использует утиную типизацию (duck-typing), в QML есть базовые типы и проверка синтаксиса при загрузке файлов.

4 QtCreator

Qt Creator (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная среда для программной разработки на C, C++ и QML. Основная задача Qt Creator — упростить разработку приложения с помощью фреймворка Qt на разных платформах. Поэтому среди возможностей, присущих любой среде разработки, есть и специфичные, такие как отладка приложений на QML и отображение в отладчике данных из контейнеров Qt, встроенный дизайнер интерфейсов как на QML, так и на QtWidgets.

Среди возможностей QtCreator стоит выделить процесс редактирования и отладки кода.

а) Редактирование кода. В Qt Creator реализовано автодополнение, в том числе ключевых слов, введённых в стандарте C++11 (начиная с версии 2.5), подсветка кода (её определение аналогично таковому в Kate, что позволяет создавать свои виды подсветок или использовать уже готовые). Также, начиная с версии 2.4, есть возможность задания стиля выравнивания, отступов и постановки скобок. Реализован ряд возможностей при работе с сигнатурами методов, а именно: автогенерация пустого тела метода после его обновления; возможность автоматически изменить сигнатуру метода в определении, если она была изменена в объявлении и наоборот; возможность автоматически поменять порядок следования аргументов. При навигации по коду доступно переключение между определением и объявлением метода, переход к объявлению метода, переименование метода как в отдельном проекте, так и во всех открытых. Также есть возможность вызвать справку согласно текущему контексту.

б) Отладка кода. Среда разработки имеет графический интерфейс для следующих отладчиков: GDB, CDB и QML/JavaScript. В качестве отдельной опции реализовано отображение содержимого контейнеров, таких как QString, std::map и прочих.

5 Реализация приложения

Целью практической части бакалаврской работы было создание независимого от платформы приложения для диспетчера транспортной службы на языках C++ и QML со следующим функционалом:

- возможность создавать многоуровневую редактируемую систему маршрутов;
- возможность сохранять, удалять и редактировать информацию о проектах в SQL базе данных;
- функции по отрисовке маршрута на карте, добавления остановок и аудио-меток объявлений;
- возможность воспроизводить и приостанавливать воспроизведение аудио-меток;
- функция экспортирования готовых маршрутов в определенном формате вместе с набором файлов, относящихся к нему;

В результате разработки была получена многоуровневая система. На верхнем уровне находится проект. Проект - это класс, представляющий собой группу маршрутов. Он содержит в себе общую для них информацию, такую как используемые языки и часовой пояс. Маршрут является основным классом приложения. В нем хранятся списки остановок в обоих направлениях, массивы отметок на карте, ссылки на звуковые файлы, координаты отрисованного на карте маршрута.

5.1 Разработка интерфейса приложения

Структурная модель представляет собой взаимодействие пользователя, рабочей станции и приложения. Весь процесс начинается с обращения пользователя через рабочую станцию к приложению, которое в свою очередь из внутренних компонентов и дополнительных библиотек формирует единый функциональный интерфейс, передаваемый пользователю. Формируемый интерфейс несколько основных типа компонентов: Кнопки, Вкладки, Поля ввода текста, Таблицы, Надписи, а также компоненты типа Карта.

Кнопки представляют собой объекты, на которые программируются основные функции приложения. Надписи отвечают за удобность и понятность интерфейса программы, подсказки и названия функций. Карта

предоставляет полный функционал по работе с географическими данными и т.д.

Главное окно приложения представляет собой прямоугольник, который является родителем для всех остальных элементов интерфейса.

При нажатии на кнопку "Создать"появляется окно создания нового проекта в режиме "создание". При двойном нажатии на проект это окно откроется в режиме "редактирование". В нем можно задать общую для всех маршрутов проекта информацию. Помимо этого, в нем описаны функции сохранения, очищения полей, а также Qt-слот, реагирующий на изменение режима запуска.

При выборе и открытии проекта пользователь попадает на главное рабочее окно приложения. Оно состоит из множества вкладок маршрутов, где каждая вкладка отвечает за один маршрут. В свою очередь, каждая вкладка маршрута делится на две части - табличное представление и карту. Карта представляет собой плагин, подключающийся к серверу с тайлами. Для удобства пользователя был добавлен режим редактирования карты. В этом режиме можно изменять или отрисовывать заново маршрут, выставлять на карту метки и остановки. В коде они описаны как mapcomponent. Помимо этого, есть возможность изменять масштаб. А также была написана модель данных, хранящая в себе все объекты, нанесенные на карту.

В табличной же части, для отображения данных используется класс inRouteDataView. Он состоит из таблицы, соответствующей ей модели данных и кнопок, позволяющих изменять содержание таблицы. Отсюда же вызываются формы для создания новых остановок, меток и добавления аудио-файлов, их редактирования. Для прослушивания аудио был написан простейший медиа-плеер на базе QtMultimedia.

5.2 Внутреннее устройство приложения

5.2.1 База Данных

Вся информация, с которой работает приложение, сохраняется в SQLite базе данных. Ее структура представлена на схеме ниже.

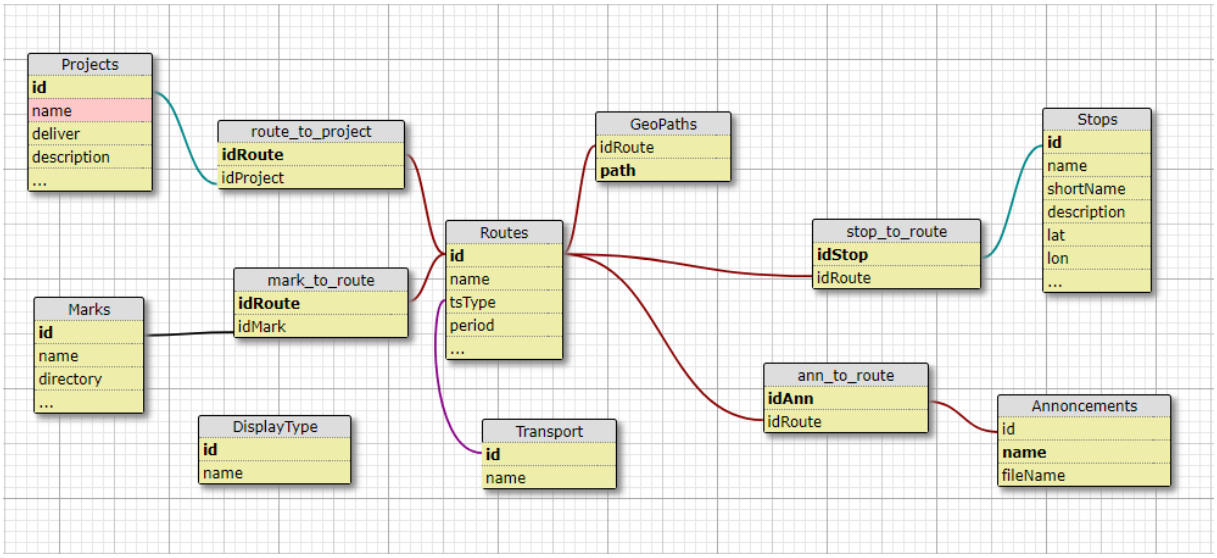

Рисунок 1 – Схема базы данных

5.2.2 Логика работы с БД

В ходе разработки приложения было написано несколько классов на C++ для реализации его внутренней логики. Для взаимодействия с базой данных было решено использовать стандартный модуль Qt 5.10.0 QtSQL. Специально для этой цели был написан класс "database".

Для хранения данных во внутренних структурах приложения был написан класс модели данных проектов - "projectModel наследуемый от стандартного класса QSQLQueryModel.

В нем происходит считывание данных из базы при помощи запроса "SELECT id, name, useEnglish, deliver, description FROM Project"и заполнение соответствующей таблицы в интерфейсе приложение. После чего заполняется сама модель данных, путем считывания функциями get-терами.

Для записи же используется метод setProject(). В нем сначала из БД удаляется проект, который будет перезаписан, затем из остальных таблиц, путем поиска соответствий через запросы, подобные "SELECT id FROM Project WHERE id = idProject в функции removeProject(int id), удаляются все его составные части, такие как маршруты, остановки, из которых они состоят, метки и другие.

После чего, если удаление прошло успешно, начинается сохранение проекта. Для этих целей была написана функция insertIntoTable(), она принимает на вход название таблицы, список данных для внесения в нее и

список форматов этих данных.

5.2.3 Основная логика

Для представления самих проектов в системе был написан класс "project". Он по сути является контейнером, в котором содержится массив маршрутов. Для реализации возможностей редактирования через интерфейс приложения, он наследуется от стандартного Qt-класса QObject. Помимо этого, в нем есть метод, позволяющий сохранять информацию в формате JSON-файла. Структура файла создается при помощи классов QJsonArray и QJsonObject. В нем сохраняется вся информация об остановках и метках, включая их географические координаты. Для представления же маршрута в системе был написан класс "route" который также является сложным контейнером и наследуется от QObject. В нем хранится массив остановок в обоих направлениях, массив меток, отрисованный на карте маршрут, ссылки на аудиофайлы. Для удобства обращения к его данным из интерфейса, некоторые массивы хранятся в виде списков формата QVariantList.

Для отображения в таблицах интерфейса были также написаны классы-моделей routeModel, содержащий в себе информацию об остановках, и audioModel, использующийся для хранения и работы с информацией об аудиофайлах. Среди их функций изменение отдельных хранящихся в них элементов, изменение их порядке при отображении, а также удаление и добавление новых элементов. Модели также являются наследниками класса QObject и дочерними классами QAbstractTableModel.

Для хранения же простейших уровней приложения были написаны классы "mark"(отметка), "stop"(остановка), "audioItem"(аудиофайл). Все они наследуются от QObject и состоят в основном из конструкторов, принимающих в качестве аргументов все необходимые данные, геттеров и сеттеров.

ЗАКЛЮЧЕНИЕ

В ходе бакалаврской работы были изучены возможности Qt 5.10.0 при создании кросс-платформенных приложений с использованием языков QML и C++ в среде QtCreator. Изучены рабочая область и функциональные возможности фреймворка Qt 5.10.0, реализованы внутренний и внешний интерфейсы приложения транспортного диспетчера для составления маршрутов, реализованы связь приложения с БД, а также реализованы инструментарии по работе с картой и мультимедийными файлами.