|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 7.3 Характеристика объекта | | | | 43 |
|  | 7.4 Мероприятия по безопасности труда и сохранению работоспособности | | | | 43 |
|  | | 7.4.1 Обеспечение требований эргономики и технической эстетики | | | 43 |
|  | | | | 7.4.1.1 Планировка помещения и размещения оборудования | 43 |
|  | | | | 7.4.1.2 цветовое оформление помещения | 47 |
|  | | | 7.4.2 Обеспечение оптимальных параметров воздуха рабочей зоны | | 48 |
|  | | | | 7.4.2.1 Нормирование параметров микроклимата | 48 |
|  | | | | 7.4.2.2 Нормирование уровней вредныхвредных хим. веществ | 49 |
|  | | | | 7.4.2.3 Нормирование уровней аэроионизации | 52 |
|  | | 7.4.3 Создание рационального освещения | | | 53 |
|  | | 7.4.4 Защита от шума | | | 53 |
|  | | 7.4.5 Обеспечение режимов труда и отдыха | | | 55 |
|  | | 7.4.6 Обеспечение электробезопасности | | | 56 |
|  | | 7.4.7 Защита от статического электричества | | | 58 |
|  | | 7.4.8 Обеспечение допустимых уровней электромагнитных поелй | | | 59 |
|  | | 7.4.9 Обеспечение пожарной безопасности | | | 60 |
|  | | | | 7.4.9.1 Обеспечениебезопасной эвакуации персонала | 62 |
|  | | | | 7.4.9.2 Средства извещения и сигнализации о пожаре | 63 |
|  | | | | 7.4.9.3 Способы и средства тушения пожара | 64 |
|  | | | | 7.4.9.4 Молниезащита объекта | 64 |
|  | 7.5 Расчет искусственного освещения | | | | 67 |
|  | 7.6 Расчет естественного освещения | | | | 69 |
| **Литература** | | | | | 72 |
| **Приложение 1. Листинг программы** | | | | | 75 |
|  | 1.1 Основная часть | | | | 75 |
|  | 1.2 Распознование | | | | 79 |
|  | 1.3 Модель таблицы | | | | 81 |
|  | 1.4 Классификация | | | | 83 |
|  | 1.5 Определение категории | | | | 85 |
|  | 1.6 Организация выпадающих списков | | | | 89 |
|  | 1.7 преобразование данных и сохранение в базу данных | | | | 90 |
|  | 1.8 расперделяемый ресурс(класс хранения данных) | | | | 95 |
|  | 1.9 Форма | | | | 96 |
|  | 1.10 Служебные классы YB.ORM | | | | 98 |
|  | 1.11 Схема базы данных | | | | 130 |
| **Приложение 2. Руководство пользователя** | | | | | 134 |

**Введение**

Проблема своевременного информирования актуальна на протяжении всего развития человечества. Например, вопрос, связанный с изменением расписания занятий. Особенно это заметно в ВУЗах, имеющих значительный контингент заочников, а также высокий процент преподавателей-совместителей.

Сейчас появилась прекрасная возможность для решения данной проблемы. В настоящее время получили самое широкое распространение достаточно мощные персональные мобильные устройства на платформе Android. По итогам третьего квартала 2012 года платформа Android заняла 75% рынка мобильных устройств. Все большее количество преподавателей ВУЗов могут воспользоваться преимуществами мобильных приложений.

Данная система доставки расписания занятий — это попытка изменить к лучшему ситуацию с информированием преподавателей в рамках кафедры, факультета, ВУЗа. Она включает в себя редактор расписания и мобильное приложение для его отображения. Однако, в процессе внедрения системы персоналу необходимо переходить с привычных средств составления расписания (MS Word, OpenOffice) на предлагаемый системой редактор, что замедляет процесс внедрения и распространения системы. Решить эту проблему позволяет разрабатываемый в данном дипломном проекте модуль для редактора расписания, позволяющий осуществить полуавтоматический ввод расписания из форматов документов docx и odt, как наиболее распространенных.

1. **Постановка задачи**

#### 1.1. Тема работы

Разработка компонента визуализации системы доставки расписания занятий.

#### 1.2. Основание для разработки

Основанием для разработки является задание на дипломный проект.

#### 1.3. Назначение и цель

*1.3.1. Назначение*

Назначением данного дипломного проекта является разработка модуля для редактора расписания системы доставки расписания занятий, который ускоряет процесс занесения данных при помощи автоматизированного ввода.

*1.3.2. Цель разработки*

Целью работы является ускорение процесса ввода данных систему доставки расписания занятий и упрощения внедрения системы в организации.

**1.4.**  **Безопасность при эксплуатации ПЭВМ**

Перед началом работы с электроизмерительными приборами пользователь должен ознакомиться с инструкцией по безопасности. Должна быть произведена классификация помещения по взрывоопасности, по степени огнестойкости, по степени поражения электрическим током.

Необходимо произвести следующие расчеты по безопасности труда:

* Расчет искусственного освещения
* Расчет естественного освещения

**1.5. Стадии и этапы разработки**

* Техническое задание.
* Техническое предложение.
* Эскизное проектирование.
* Техническое проектирование.
* Рабочая документация.

#### 1.6. Порядок контроля и приемки

Проверка на работоспособность должна осуществляться последовательной проверкой выполнения всех встроенных функций после установки данного продукта. Работа должна быть принята руководителем дипломного проекта.

1. **Требования к функциональным характеристикам**

**2.1. Требования к программе:**

- Считывать и распознавать расписание из файла

- Классифицировать полученные данные

- Предоставлять пользователю возможность отредактировать данные

- Сохранять данные в базе данных

**2.2. Требования к программному обеспечению:**

- Операционная система Windows XP или выше, Linux, Mac OS

- Драйвер ODBC

**2.3. Требования к аппаратным характеристикам:**

- Не менее 512Mб RAM

- 100Мб свободного места на жестком диске

#### 2.4. Требования к документации

К программному изделию должны прилагаться следующие документы:

- Техническое описание.

- Руководство пользователя.

**2.5. Требования к безопасности**

Аппаратные средства должны быть заземлены в соответствии с инструкцией по эксплуатации ПК.

**2.6. Требования к надежности**

Срок службы модуля импорта не ограничен.

**3. Выбор средств реализации**

Так как редактор расписания написан на языке программирования С++ с использованием библиотек Qt, то для обеспечения наилучшей совместимости для модуля был выбран язык С++ и использован набор библиотек Qt. Для работы с БД используется проект YB.ORM, который не имеет доступных аналогов с заданной функциональностью.

**3.1. С++**

**C++** — [компилируемый](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) [статически типизированный](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) общего назначения.

Поддерживает такие [парадигмы программирования](http://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D1%8B_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как [высокоуровневых](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), так и [низкоуровневых языков](http://ru.wikipedia.org/wiki/%D0%9D%D0%B8%D0%B7%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). В сравнении с его предшественником — языком [C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), — наибольшее внимание уделено поддержке [объектно-ориентированного](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [обобщённого программирования](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание [операционных систем](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), разнообразных прикладных программ, [драйверов](http://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений.

Синтаксис C++ унаследован от языка [C](http://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)). Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как [компиляторами](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80) C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

**3.2 Qt**

Qt — [кросс-платформенный](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Кроссплатформенное программное обеспечение) инструментарий разработки ПО на языке программирования [C++](http://ru.wikipedia.org/wiki/C%2B%2B).

Позволяет запускать написанное с его помощью [ПО](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в большинстве современных [операционных систем](http://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) путём простой компиляции программы для каждой ОС без изменения [исходного кода](http://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Включает в себя все основные [классы](http://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые могут потребоваться при разработке [прикладного программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), начиная от элементов [графического интерфейса](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и заканчивая классами для работы с [сетью](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), [базами данных](http://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и [XML](http://ru.wikipedia.org/wiki/XML). Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Отличительная особенность Qt от других библиотек — использование *[Meta Object Compiler](http://ru.wikipedia.org/w/index.php?title=%D0%9C%D0%B5%D1%82%D0%B0%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80&action=edit&redlink=1" \o "Метаобъектный компилятор (страница отсутствует)) (MOC)* — предварительной системы обработки исходного кода (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как *слоты* и *сигналы*. Кроме того, это позволяет сделать код более лаконичным. Утилита MOC ищет в заголовочных файлах на [C++](http://ru.wikipedia.org/wiki/C%2B%2B) описания классов, содержащие макрос Q\_OBJECT, и создаёт дополнительный исходный файл на [C++](http://ru.wikipedia.org/wiki/C%2B%2B), содержащий *метаобъектный* код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности [виджетов](http://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B6%D0%B5%D1%82" \o "Виджет), связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

**3.3. YB.ORM**

Инструмент YB.ORM разрабатывается для упрощения создания приложений на C++, которые имеют дело с базами данных SQL. Цель - предоставить удобный интерфейс наподобие SQLAlchemy (Python) или Hibernate (Java). Доступны порты для Linux и Windows (MinGW, VC). Поддерживаемые диалекты SQL: MySQL, Postgres, Oracle, Firebird, SQLite. Типичный сценарий использования:

* Опишите схему базы данных, т.е. таблицы и их колонки, а также связи между ними в простом формате, основанном на XML.
* Опционально, используйте генератор SQL DDL для создания скрипта с DDL. Выполните этот шаг, чтобы наполнить пустую схему БД таблицами.
* Используйте инструмент для генерации кода, создающий классы C++ для каждой таблицы. Сгененрированный код скрывает детали того, как объекты загружаются и сохраняются из/в таблицы БД, также он представляет отношения между объектами в удобном виде.
* Добавьте в классы бизнес-логику приложения, и т.д.
* Когда вы внесёте изменения в описание схемы, можно будет использовать инструмент для генерации кода чтобы синхронизировать схему с ранее сгенерированными классами С++.
* Теперь вы можете использовать сгенерированные классы в связке с объектом Session для автоматизации следующих задач:
  + запрашивать объекты из базы данных,
  + создавать новые объекты или
  + загружать и изменять существующие объекты, и сохранять их в базу,
  + связывать и развязывать объекты друг к/от другу, используя отношения,
  + удалять объекты из базы данных.

**3.4. Система управления версиями**

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Наиболее распространенными на данный момент системами контроля версиями являются: GIT, SVN и Mercurial.

**3.3.1. GIT**

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года. На сегодняшний день поддерживается Джунио Хамано.

Примерами проектов, использующих Git, являются ядро Linux, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki и некоторые дистрибутивы Linux (см. ниже).

Программа является свободной и выпущена под лицензией GNU GPL версии 2.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером фронтенда к репозиториям Git, а StGit использует Git для управления коллекцией патчей.

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone, Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

**3.3.2. SVN**

Subversion (также известная как «SVN») — свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet Inc.

Цель проекта — заменить собой распространенную на тот момент систему Concurrent Versions System (CVS), которая ныне считается устаревшей. Subversion реализует все основные функции CVS и свободна от ряда недостатков последней.

В настоящее время Subversion используется многими сообществами разработчиков открытого программного обеспечения (в том числе сообществами, ранее использовавшими CVS). В их числе такие известные проекты, как Apache, GCC, Free Pascal, Python, Ruby, FreeBSD, AROS, Blender, Boost, Tor, OGRE. Subversion также широко используется в закрытых проектах и корпоративной сфере. Хостинг Subversion, в том числе для проектов с открытым кодом, также предоставляют популярные хостинг-проекты SourceForge.net, Tigris.org, Google Code и BountySource.

В 2007 году независимая компания Forrester Research, сравнивая преимущества и недостатки различных систем, оценила Subversion как «единоличного лидера в категории Standalone Software Configuration Management (SCM) и сильного участника в категории Software Configuration and Change Management (SCCM)».

По данным статистики использования пакетов Linux-дистрибутивов Debian и Ubuntu, количество активных пользователей Subversion примерно такое же, как у Git, и превосходит аналогичный показатель для CVS, Mercurial и Bazaar (по состоянию на июнь 2011 года)

Subversion — централизованная система (в отличие от распределённых систем, таких как Git или Mercurial), то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере.

Работа в Subversion мало отличается от работы в других централизованных системах управления версиями. Клиенты копируют файлы из хранилища, создавая локальные рабочие копии, затем вносят изменения в рабочие копии и фиксируют эти изменения в хранилище. Несколько клиентов могут одновременно обращаться к хранилищу. Для совместной работы над файлами в Subversion преимущественно используется модель копирование — изменение — слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель блокирование — изменение — разблокирование.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

При использовании доступа с помощью WebDAV также поддерживается прозрачное управление версиями — если любой клиент WebDAV открывает для записи и затем сохраняет файл, хранящийся на сетевом ресурсе, то автоматически создаётся новая версия.

**3.3.3. Mercurial**

Mercurial (англ. ртутный, подвижный), он же Hg (от обозначения химического элемента ртути) — кроссплатформенная распределенная система управления версиями, разработанная для эффективной работы с очень большими репозиториями кода. В первую очередь она является консольной программой.

Система Mercurial написана на Python, хотя чувствительные к производительности части (например, своя реализация diff) выполнены в качестве модулей-расширений на C. Mercurial первоначально была написана для Linux, позже портирована под Windows, Mac OS X и большинство Unix-систем. Репозитории Mercurial управляются при помощи утилиты командной строки hg, но есть и GUI–интерфейсы.

Наряду с традиционными возможностями систем контроля версий, Mercurial поддерживает полностью децентрализованную работу (отсутствует понятие основного хранилища кода), ветвление (возможно вести несколько веток одного проекта и копировать изменения между ветками), слияние репозиториев (чем и достигается «распределённость» работы). Поддерживается обмен данными между репозиториями через HTTP/HTTPS, SSH и вручную при помощи упакованных наборов изменений.

Утилита hg обладает компактным интерфейсом, и Mercurial считается более простой в освоении системой, чем, например, git.

Для использования в данном проекте была выбрана система GIT. Это связано, во-первых, с ее децентрализованностью (проект является открытым и предоставляет любому программисту возможность заниматься его разработкой), а, во-вторых, с множеством бесплатных хостингов репозиториев данного типа (GitHub, GoogleCode, SourgeForce), предоставляющих множество возможностей для совместной разработки и анализа статистики. Так же GIT обладает наибольшей скорость работы относительно всех перечисленных систем, так как полностью написана на С.

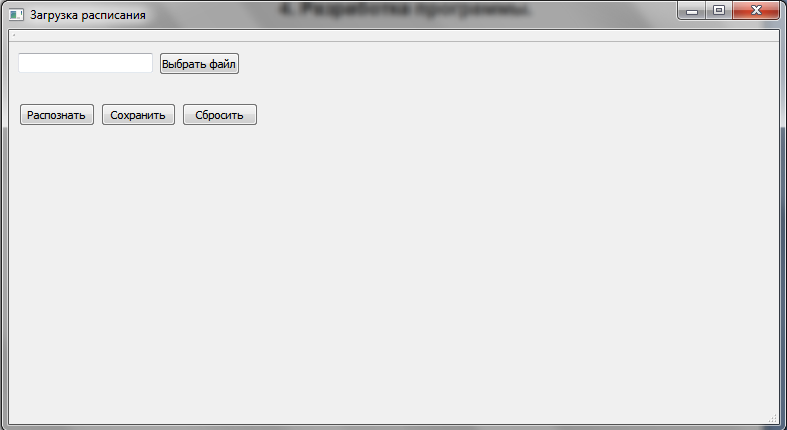
1. **Разработка программы**

**4.1. Разработка пользовательского интерфейса**

Разработка простого и доступного пользовательского интерфейса является одной из основных задач, так как именно от этого зависит успех взаимодействия пользователя и ЭВМ. Простота и интуитивность интерфейса в значительной мере позволяет сократить время, которое потребуется пользователю для выполнения операций.

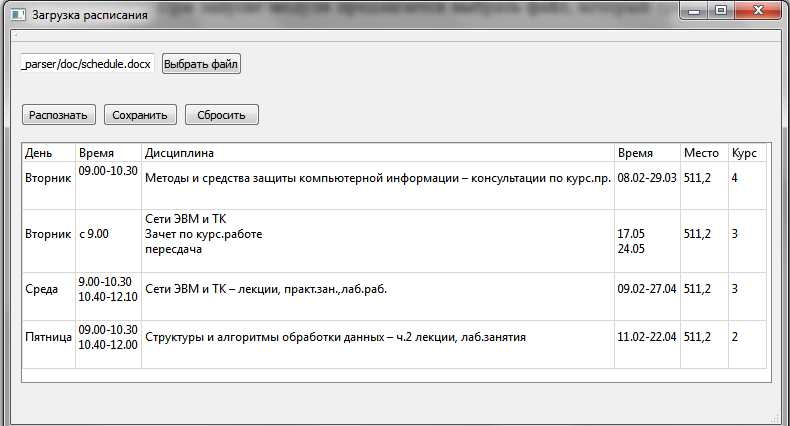
При запуске модуля появляется экранная форма:

Рисунок 4.1.1.



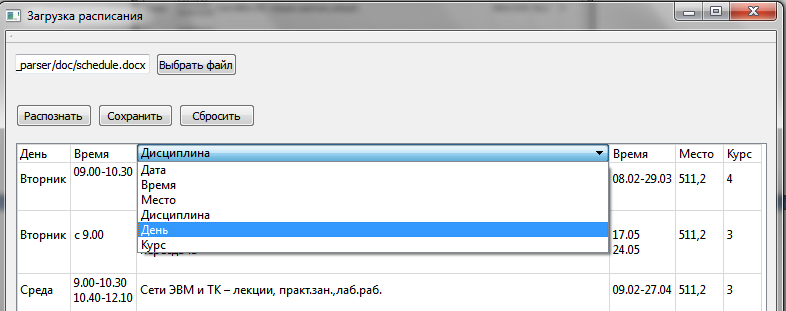
На форме предлагается выбрать файл, который требуется распознать. Для этого нажимается кнопка «Выбрать файл» и нужно выбрать его. Файлы можно выбирать только с расширением \*.docx или \*.odt. После выбора нужного файла требуется нажать на кнопку «Распознавать» и модуль начнет свою работу. На выходе будет представлена таблица:

Рисунок 4.1.2.



Если в таблице заголовочные ячейки отображены не верно, их можно поправить, используя выпадающий список и выбрать нужное:

Рисунок 4.1.3.



Конечный результат сохраняется при помощи кнопки « Сохранить» в базу данных.

Если требуется начать работу заново или сбросить все параметры используется кнопка «Сбросить», которая удаляет таблицу с формы, однако путь остается на месте и можно выбрать тот же самый файл.

**4.2. Реализация программы**

Для реализации поставленной задачи используется язык программирования C++, библиотеки Qt и среда разработки Qt Creator.

Соединение сигналов элементов формы со слотами. Это необходимо для задания поведению элементов, расположенных на форме (кнопок и т.п.):

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow){

ui->setupUi(this);

this->setWindowTitle(\_R("Загрузка расписания")); //заголовок окна

QObject::connect(ui->changeFileButton, SIGNAL(clicked()), this, SLOT(setFilePath())); //выбор файла

QObject::connect(ui->parseButton, SIGNAL(clicked()), this, SLOT(parseFile())); // распознавание

}

Дальнейшее развитие программы происходит при нажатии на кнопки. Всего их четыре:

1. При нажатии на кнопку «Выбрать файл» (changeFileButton) вызывается функция setFilePath. Она открывает диалоговое окно выбора файла с помощью встроенных в ОС средств и записывает путь к файлу в окно ввода текста filePathEdit:

void

MainWindow::setFilePath()

{

QString filePath = QFileDialog::getOpenFileName(0, \_R("Файл расписания"), "", "\*.docx");

ui->filePathEdit->setText(filePath);

}

При нажатии на кнопку появляется диалоговое окно, в котором предлагается осуществить выбор файла, из которого будут извлекаться данные. Файлы можно выбирать только с расширением \*.docx или \*.odt.

1. При нажатии на кнопку «Сбросить» (clearButton) вызывается функция clearTable:

MainWindow::clearTable()

{

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

QLayoutItem \*item = layout->*takeAt*(0);

if (item)

{

QWidget \*widget = item->*widget*();

delete item;

layout->removeWidget(widget);

delete widget;

}

}

Данная функция полностью удаляет виджет таблицы с формы.

1. При нажатии на кнопку «Сохранить» (saveButton) вызывается функция saveTable:

void

MainWindow::saveTable()

{

QVBoxLayout \*layout = (QVBoxLayout \*) ui->infoLayout;

QComboBox \*cb = (QComboBox \*) layout->*takeAt*(0)->*widget*();

Yb::Session session(Yb::theMetaData::instance(), SharedObjSingleton::instance().getEngine());

qDebug() << "Name: " << cb->currentText();

Domain::User user = Yb::query<Domain::User>(session)

.filter\_by(Domain::User::c.name == cb->currentText()).one();{

QStringList categories;

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

QTableView \*table = (QTableView \*) layout->*itemAt*(0)->*widget*();

QAbstractItemModel \*model = table->model();

for (int i = 0; i < model->*columnCount*(); ++i){

QModelIndex index = model->*index*(0, i, QModelIndex());

categories << index.data().toString();

}

SharedObjSingleton::instance().setCategories(categories);

}

ScheduleManager scheduleManager(&user);

scheduleManager.convertToScheduleObj();

QString msg = \_R("Расписание сохранено для пользователя: ");

msg += user.name.value();

QMessageBox::information(0, \_R("Радость"), msg);

}

Происходит соединение с базой данных, из которой запрашивается выбранный пользователь. Далее в цикле выбираются все заголовки столбцов и заносятся в массив categories, который по выходу из цикла сохраняется в класс общих ресурсов. Создается экземпляр класса ScheduleManager, который производит преобразование данных к объектам Schedule и сохраняет их в базу данных. Структура класса:

class ScheduleManager

{

typedef void(ScheduleManager::\*Handler)(int, int);

Yb::Session \*session\_;

Domain::User \*user\_;

QMap<QString, Handler> handlers\_;

QStringList categories\_;

QVector<Row> table\_;

bool specialCaseProcessed\_;

Domain::ScheduleHolder \*scheduleHolder\_;

void dowHandler(int row, int column);

void subjectHandler(int row, int column);

void placeHandler(int row, int column);

void timeHandler(int row, int column);

void dateHandler(int row, int column);

void courseHandler(int row, int column);

public:

ScheduleManager(Domain::User \*user);

void convertToScheduleObj(bool save = false);

~ScheduleManager(){

delete session\_;

}

};

Класс содержит в себе набор функций, которые преобразовывают колонку определенной категории. Функция convertToScheduleObj() обходит в цикле все строки и столбцы таблицы, вызывая соответствующую функцию-обработчик для каждой ячейки. После каждой итерации в цикле строк получается готовый объект Schedule, который сохраняется в базу данных.

1. При нажатии на кнопку «Распознать» (parseButton) вызывается функция parseFile:

void

MainWindow::parseFile()

{

QString filePath = ui->filePathEdit->text();

if (filePath == "")

{

QMessageBox::information(0, \_R("Ошибка"), \_R("Файл не выбран"));

return;

}

QZipReader zip(filePath);

if (zip.exists())

{

try {

QByteArray content = zip.fileData("word/document.xml");

DocumentParser parser(content);

QVector<Row> rows = parser.getRows();

QTableView \*table = new QTableView();

TableModel \*model = new TableModel(rows);

table->*setModel*(model);

table->setItemDelegateForRow(0, new TableHeaderDelegate);

table->resizeColumnsToContents();

table->verticalHeader()->setResizeMode(QHeaderView::ResizeToContents);

table->horizontalHeader()->hide();

table->verticalHeader()->hide();

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

clearTable();

layout->addWidget(table);

QPushButton \*clearButton = new QPushButton();

clearButton->setObjectName("clearButton");

clearButton->setText(\_R("Очистить"));

QObject::connect(clearButton, SIGNAL(clicked()), this, SLOT(clearTable()));

QPushButton \*saveButton = new QPushButton();

saveButton->setObjectName("saveButton");

saveButton->setText(\_R("Сохранить"));

QObject::connect(saveButton, SIGNAL(clicked()), this, SLOT(saveTable()));

QStringList users = getUsers();

QComboBox \*cb = new QComboBox();

cb->addItems(users);

cb->setObjectName("userComboBox");

layout = (QVBoxLayout \*) ui->infoLayout;

layout->addWidget(cb);

layout = (QVBoxLayout \*) ui->buttonLayout;

layout->addWidget(saveButton);

layout->addWidget(clearButton);

}

catch(ParserEx)

{

QMessageBox::information(0, \_R("Ошибка"), \_R("Документ поврежден"));

}

}

zip.close();

}

Вначале программа считывает путь к файлу из окна ввода текста filePathEdit в переменную filePath. После чего проверяет, выбран ли файл и, если нет, выдает ошибку:

QString filePath = ui->filePathEdit->text();

if (filePath == "")

{

QMessageBox::information(0, \_R("Ошибка"), \_R("Файл не выбран"));

return;

}

Далее стандартной функцией Qt разархивируем предлагаемый файл:

QZipReader zip(filePath);

Проверяем, все ли прошло успешно и, если нет, возвращаем ошибку «Файл поврежден». Иначе выбираем нужный нам файл document.xml и создаем экземпляр класса DocumentParser parser, передавая извлеченный файл в качестве аргумента конструктора:

DocumentParser::DocumentParser(const QByteArray &content, const QString &doctype)

{

QDomDocument doc;

doc.setContent(content, false);

if (doctype == "docx")

{

QDomNodeList tables = doc.elementsByTagName("w:tbl");

if (tables.size() == 0)

throw ParserEx("DocumentParser", "no one table found in document");

table\_ = tables.at(0);

// tr обозначает строку

rows\_ = table\_.toElement().elementsByTagName("w:tr");

}

}

Ищем корневой элемент разметки таблицы "w:tbl" и, если не находим, возвращаем сообщение о том, что в документе не найдено ни одной таблицы. В противном случае находим все строки в таблице.

Возвращаемся к функции parseFile и вызываем у переменной parser метод getRows():

QVector<Row> rows = parser.getRows();

На выходе получаем двумерный массив, состоящий из массива строк:

QVector<Row>

DocumentParser::getRows()

{

QVector<Row> rows\_vector;

for (int i = 0; i < rows\_.count(); ++i)

{

// tc - столбец

QDomNodeList cells = rows\_.at(i).toElement().elementsByTagName("w:tc");

Row row;

for (int j = 0; j < cells.count(); ++j)

{

// p - блок с записью (предположительно до перевода строки)

QDomNodeList blocks = cells.at(j).toElement().elementsByTagName("w:p");

Cell cell;

for (int k = 0; k < blocks.count(); ++k){

QString content = parseBlock(blocks.at(k));

cell.addBlock(content);

}

row.push\_back(cell);

}

rows\_vector.push\_back(row);

}

return rows\_vector;

}

Создаем цикл по строкам, находим все столбцы в текущей строке и образуем еще один цикл, уже из найденных столбцов.

Во внутреннем цикле распознаются блоки текста. Используется функция parseBlock:

QString

DocumentParser::parseBlock(QDomNode block){

QString textInBlock = "";

// В t текстовое содержимое

QDomNodeList contents = block.toElement().elementsByTagName("w:t");

for (int i = 0; i < contents.count(); ++i){

textInBlock += contents.at(i).toElement().text();

}

return textInBlock;

}

Такая организация функции необходима, так как визуально однородный текст в ячейке таблицы может находиться в разных блоках в разметке документа. Это связано с разными стилями в одной записи, шрифтами, языками и т.п.

Так же в этом цикле все найденные блоки записываются в массив.

Во внешнем цикле полученный массив (суть отражающий строку таблицы) записывается в новый массив, который и вернет функция.

Далее в функции parseFile создаем виджет таблицы и модель таблицы.

QTableView \*table = new QTableView();

TableModel \*model = new TableModel(rows);

Вызывается конструктор TableModel:

TableModel::TableModel(QVector<Row> table, QObject \*parent) :

QAbstractTableModel(parent)

{

table.remove(0);

SharedObjSingleton::instance().setRows(table);

CategoryFactory \*factory = new CategoryFactory();

AbstractCategory \*category = new DowCategory();

factory->addCategory(category);

category = new SubjectCategory();

factory->addCategory(category);

category = new DateCategory();

factory->addCategory(category);

category = new TimeCategory();

factory->addCategory(category);

category = new PlaceCategory();

factory->addCategory(category);

category = new CourseCategory();

factory->addCategory(category);

Classifier cf(table, factory);

table\_ = cf.getRows();

table\_.push\_front(cf.suggestionParse());

}

Удаляется шапка таблицы. Создаются категории. Категории являются классами, имеющими один единственный метод – рассчитывающий по определенным правилам вероятность, с которой переданный ему столбец принадлежит данной категории. Создается фабрика категорий (объект класса CategoryFactory – класс реализует паттерн Абстрактная Фабрика, позволяющий хранить множество различных объектов – наследников одного класса), категории заносятся в фабрику. Создается классификатор (Classifier cf), в качестве аргументов которому передается двумерный массив и фабрика категорий. В конструкторе классификатор приводит строки к виду, пригодном для отображения в таблице. Мы получаем их с помощью метода getRows(). Далее вызываем метод классификации столбцов suggestionParse():

QVector<QString> headerList;

for (int i = 0; i < source\_table\_.at(0).size(); ++i){

QVector<Cell> currentColumn;

ColumnStatistic statistic;

for (int j = 0; j < source\_table\_.size(); ++j){

Cell currentCell = source\_table\_.at(j).at(i);

calculateCharacteristic(currentCell, statistic);

currentColumn.push\_back(currentCell);

}

statistic.normalize(source\_table\_.size());

QMultiMap<double, AbstractCategory\*> rez = categories\_->calculateProbability(statistic, currentColumn);

QString header = (rez.end()-1).value()->getName();

headerList.push\_back(header);

}

return headerList;

}

Вложенные циклы, сначала по столбцам, потом по строкам. Во внутреннем вызывается метод calculateCharacteristic, функция в котором считаются параметры для ячеек текущего столбца.

void

Classifier::calculateCharacteristic(const Cell &cell, ColumnStatistic &statistic){

foreach (QString str, cell.blocks\_){

if (str.size() > statistic.maxStringSize)

statistic.maxStringSize = str.size();

if (str.size() < statistic.minStringSize)

statistic.minStringSize = str.size();

QStringList words = str.split(" ");

statistic.words += words.size();

foreach (QChar ch, str){

if (ch == QChar(','))

statistic.hasComa = true;

if (ch == QChar(':'))

statistic.hasCodon = true;

if (ch == QChar('-'))

statistic.hasDash = true;

if (ch.isDigit())

statistic.numbers++;

if (ch.isLetter())

statistic.symbols++;

}

}

}

В этой функции считаются максимальная и минимальная длина строки, количество букв, цифр и слов (по пробелам), а так же присутствует ли в столбце символ запятой и спецсимволы (дефис, двоеточие).

Все ячейки так же заносятся в массив. В итоге на выходе внутреннего цикла получается структура со статистикой и массив ячеек, отражающий только что пройденный столбец.

Во внешнем цикле функции классификатора вызывается функция нормализации (normalize):

void normalize(int count)

{

words = words/count;

symbols = symbols/count;

numbers = numbers/count;

}

Где данные статистики, полученные ранее с помощью метода calculateCharacteristic, нормализуются путем вычисления среднего арифмитического.

После чего все полученные данные отправляются в функцию calculateProbability фабрики категорий:

QMultiMap<double, AbstractCategory\*>

CategoryFactory::calculateProbability(ColumnStatistic &statistic, QVector<Cell> &column)

{

QMultiMap<double, AbstractCategory\*> probability;

foreach (AbstractCategory \*category, category\_)

{

double prob = category->*calculateProbability*(statistic, column);

probability.insert(prob, category);

}

return probability;

}

Здесь происходит цикл по всем категориям. У каждой категории вызывается метод calculateProbability, возвращающий вероятность принадлежности столбца данной категории. Полученные данные складываются в ассоциативный массив, где ключом служит вероятность, а значением – категория. Получившийся массив возвращается из функции в suggestionParse.

Так как ассоциативный массив является упорядоченным по возрастанию, возвращаем имя категории, которая находится последней в массиве (это будет категория с наибольшей вероятностью).

Запоминаем шапку для всех столбцов:

headerList.append(headerString);

Настраиваем таблицу:

table->*setModel*(model);

table->setItemDelegateForRow(0, new TableHeaderDelegate); //выпадающий список

table->resizeColumnsToContents(); //растягивает

table->verticalHeader()->setResizeMode(QHeaderView::ResizeToContents); //растягивает

table->horizontalHeader()->hide(); //удаляет номера столбцов

table->verticalHeader()->hide(); //удаляет номера строк

Получаем ссылку на область разметки и добавляем в нее виджет таблицы:

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

clearTable();

layout->addWidget(table);

Добавляем на форму дополнительные элементы: кнопки «Очистить», «Сохранить», соединяем сигналы с соответствующими слотами.

QPushButton \*clearButton = new QPushButton();

clearButton->setObjectName("clearButton");

clearButton->setText(\_R("Очистить"));

QObject::connect(clearButton, SIGNAL(clicked()), this, SLOT(clearTable()));

QPushButton \*saveButton = new QPushButton();

saveButton->setObjectName("saveButton");

saveButton->setText(\_R("Сохранить"));

QObject::connect(saveButton, SIGNAL(clicked()), this, SLOT(saveTable()));

QStringList users = getUsers();

QComboBox \*cb = new QComboBox();

cb->addItems(users);

cb->setObjectName("userComboBox");

layout = (QVBoxLayout \*) ui->infoLayout;

layout->addWidget(cb);

layout = (QVBoxLayout \*) ui->buttonLayout;

layout->addWidget(saveButton);

layout->addWidget(clearButton);

Так же на форму добавляется список пользователей в виде выпадающего списка. Сам список получает из базы данных функция getUsers():

QStringList

MainWindow::getUsers()

{

QStringList userList;

Yb::Session session(Yb::theMetaData::instance(), SharedObjSingleton::instance().getEngine());

Yb::DomainResultSet<Domain::User> users = Yb::query<Domain::User>(session).all();

Yb::DomainResultSet<Domain::User>::iterator ibegin = users.begin(), iend = users.end();

for (; ibegin != iend; ++ibegin)

{

userList << ibegin->name.value();

}

return userList;

}

Класс, позволяющий превратить ячейки таблицы в выпадающий список. Данный класс называется Делегатом и служит как бы прослойкой между представлением и непосредственно моделью таблицы.

class TableHeaderDelegate : public QItemDelegate

{

public:

TableHeaderDelegate(QObject \*parent = 0);

QWidget\* *createEditor*(QWidget \*parent, const QStyleOptionViewItem &option,

const QModelIndex &index) const;

void *setEditorData*(QWidget \*editor, const QModelIndex &index) const;

void *setModelData*(QWidget \*editor, QAbstractItemModel \*model,

const QModelIndex &index) const;

};

Определение конструктора:

QWidget\*

TableHeaderDelegate::*createEditor*(QWidget \*parent, const QStyleOptionViewItem &option,

const QModelIndex &index) const

{

QComboBox \*combo\_box = new QComboBox(parent);

QStringList categories;

categories << \_R("Дата") << \_R("Время") << \_R("Место") << \_R("Дисциплина") << \_R("День") << \_R("Курс");

combo\_box->addItems(categories);

return combo\_box;

}

Здесь происходит создание выпадающего списка и наполнения его данными. Конструктор вызывается при двойном щелчке мышью по соответствующей ячейке таблицы. Метод setModelData() вызывается при выборе значения в списке и меняет данные модели таблицы:

void

TableHeaderDelegate::*setModelData*(QWidget \*editor, QAbstractItemModel \*model,

const QModelIndex &index) const

{

QComboBox \*combo\_box = qobject\_cast<QComboBox \*>(editor);

QVariant val = combo\_box->currentText();

model->*setData*(index, val, Qt::EditRole);

}

**5. Тестирование программы**

На каждом этапе разработки проект тестировался для выявления ошибок и проверки корректности работы программы. Основным видом тестирования, которое проводилось над данным модулем, является функциональное тестирование. Функциональное тестирование — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

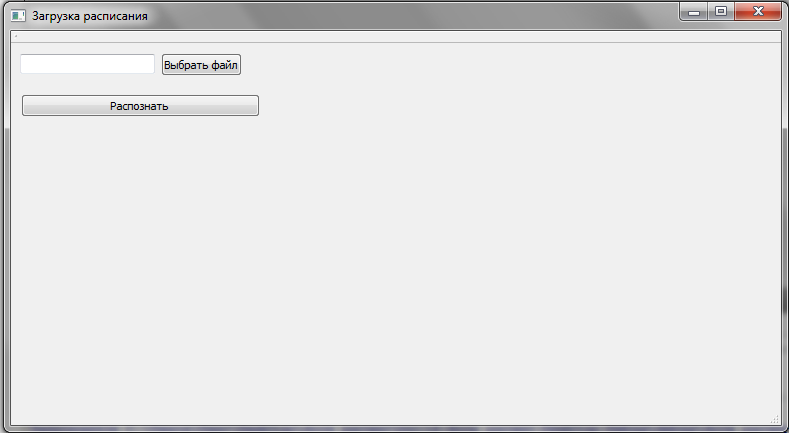
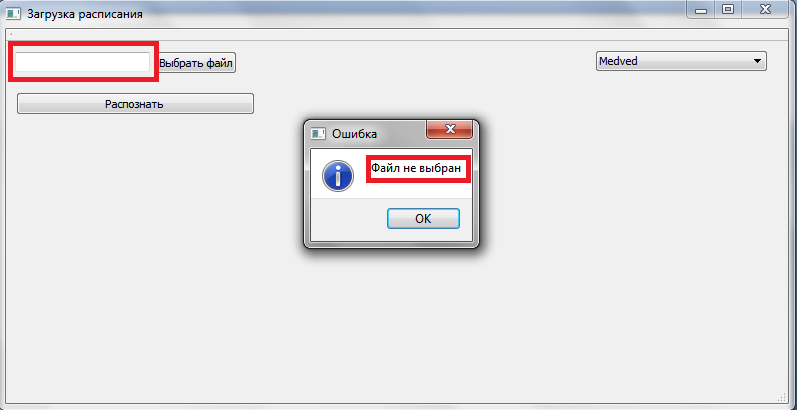
При запуске данного модуля, первое, что видит пользователь - это форма для импорта расписания.

Рисунок 5.1.

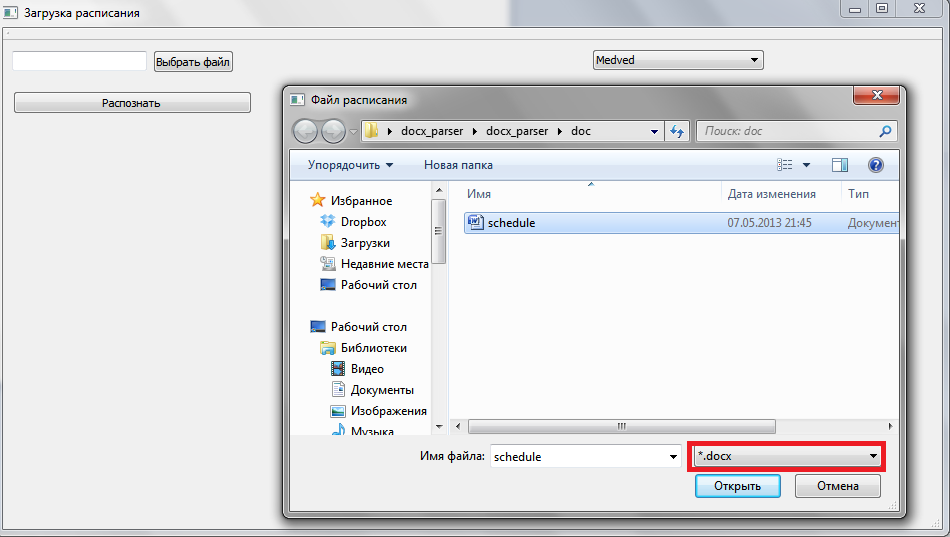
Суть модуля в импорте файла с таблицей в формате \*.docx или \*.odt в базу данных для дальнейшей работы с данными. На форме есть окно для ввода и две кнопки. Для того чтобы импортировать необходимо подать файл на вход программы. После чего нажать кнопку «Распознать» для дальнейших действий. Если оставить поле пустое и сразу нажать «Распознать» , возникает ошибка и выскакивает сообщение - предупреждение для пользователя.

Рисунок 5.2.



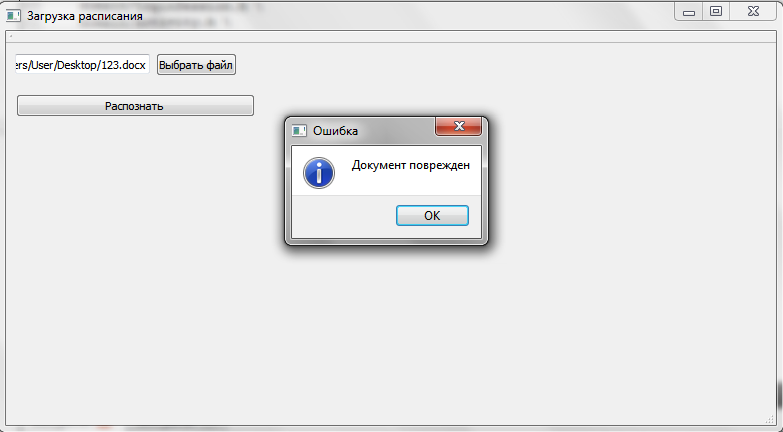
В списке файлов будут представлены только поддерживаемые расширения файлов.

Рисунок 5.3.



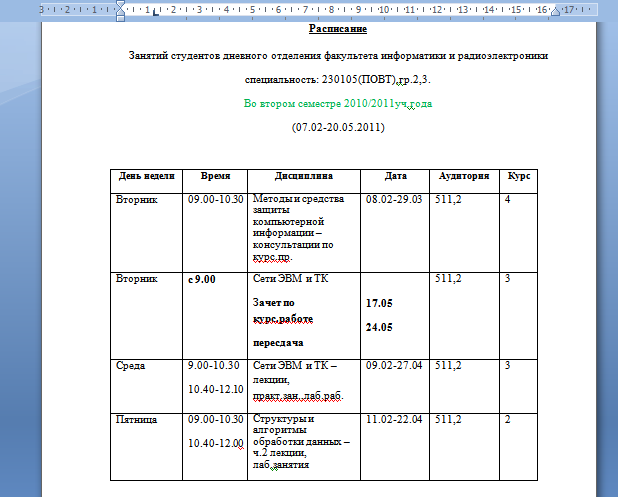
Файл должен содержать таблицу, если в файле нет таблицы, программа выдаст ошибку с сообщением-предупреждением «Документ поврежден», так как подразумевается, что все файлы поданные на вход должны содержать таблицу с данными для импорта.

Рисунок 5.4.



Пример входного файла с расширением \*.docx:

Рисунок 5.5.



Поданный на вход файл разархивируется на множество файлов в формате xml. Для входного файла из примера с расширением \*.docx ищется документ document.xml, а уже в нем распознается корневой элемент таблицы, в случае его отсутствия и выдается ошибка «документ поврежден».

Тег для начала таблицы это tbl. В файле с примером он выглядит так:

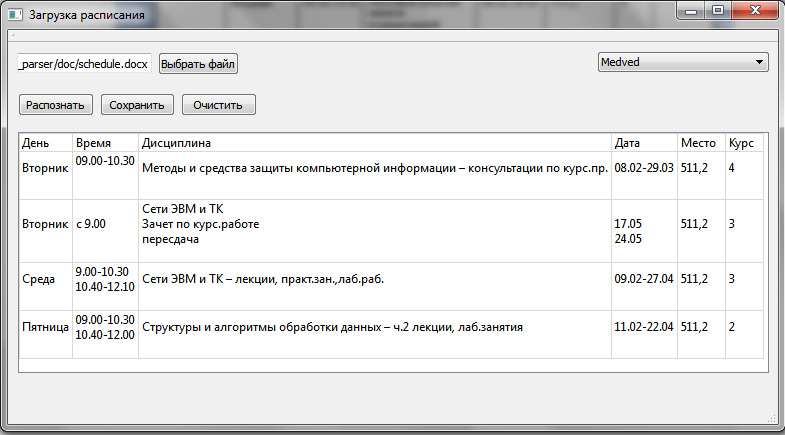
Рисунок 5.6.



Ячейки таблицы располагаются под тегами tr (строка) и tc (столбец). Блоки с текстом находятся под тегом p. Используя эти теги программа распознает блоки текстов и позиции этих блоков относительно строк и столбцов.

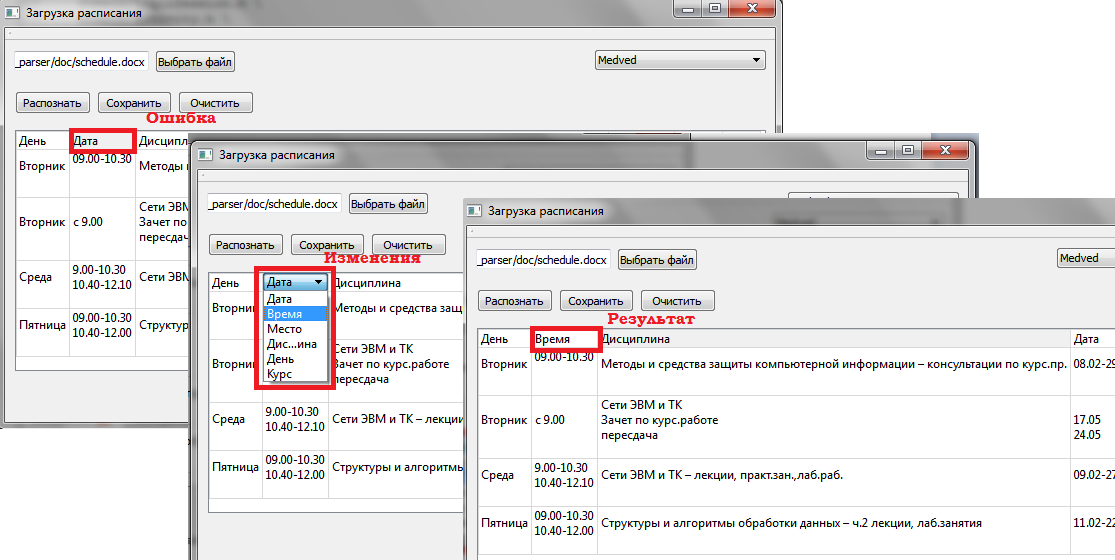
После программа производит анализ полученных данных и классифицирует их по категориям, чтобы потом вывести на экран полученную информацию для сохранения в базу данных или для корректировки в случае неверных результатов обработки.

Рисунок 5.7.

****

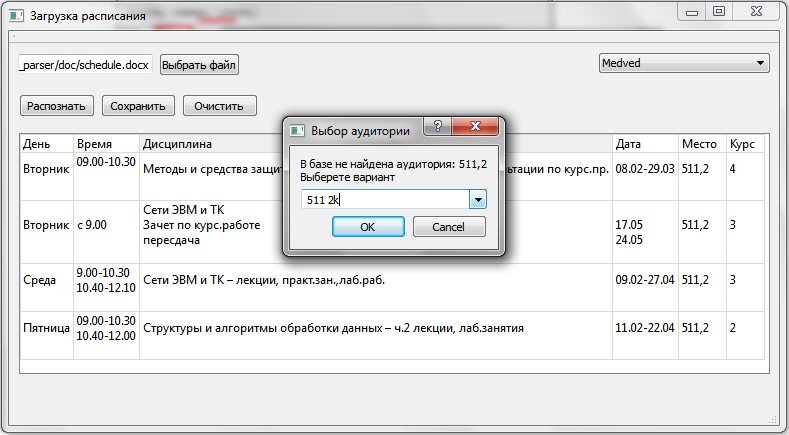
Если полученная информация не соответствует ожидаемым требованиям ее можно и нужно отредактировать. Изменения в нужную графу вносят при двойном нажатии на ячейку, после чего выпадает список возможных вариантов для исправления.

Рисунок 5.8.



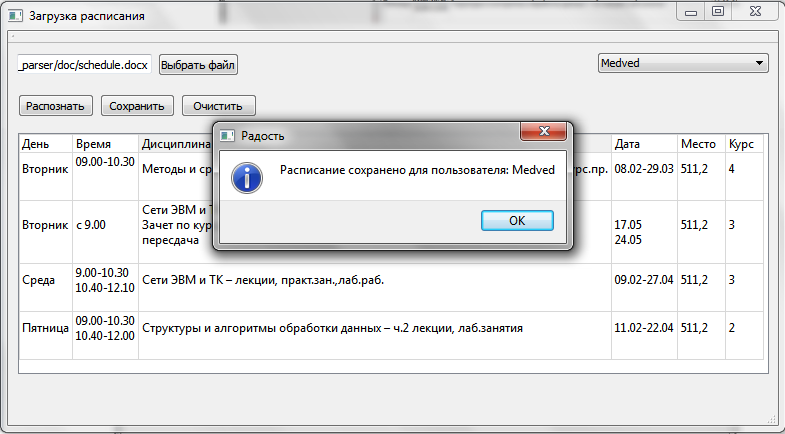
После корректирования, если это требовалось, следует сохранить расписание в базу данных для дальнейшей работы с ней. При внесении в базу могут возникнуть ошибки связанные с особенностями обозначения данных.

Рисунок 5.9.



В случае, если в базе такого значения нет, его можно добавить. При успешном сохранении выдается оповещение. После чего можно продолжить работу со следующими файлами.

Рисунок 5.9.



**6. Организационно-экономическая часть**

**6.1.Технико-экономическое обоснование проекта.**

Целью работы является с разработка компонента визуализации системы доставки расписания занятий. Проект является отдельным независимым модулем системы. Задача проекта заключается в автоматизировании ввода данных в редактор расписания.

Данный модуль позволяет значительно сократить время и трудозатраты на обработку и представление информации.

Так как проект не носит комерческого характера задача на получение дохода не ставится.

**6.2.Организационная часть.**

Для организации выполнения поставленной задачи необходимо определить группу разработчиков.

Таблица 6.1. Состав группы разработчиков и их должностные оклады.

|  |  |  |
| --- | --- | --- |
| Категория исполнителей | Количество чел. | Должностной оклад |
| Ведущий инженер  Инженер-программист | 1  2 | 30 000  15 000 |

В соответствии с ГОСТ необходимо распределить между разработчиками работы по выполнению проекта и определить трудозатраты в днях для каждого участника группы разработчиков.

*Таблица 6.2. Этапы и работы по разработке программного обеспечения.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Этап | Содержание работ,  входящих в этап | Количество  исполнителей | | Продолжи-тельность  работ, дни |
| чел. | должность |
| Разработка технического задания | Постановка задачи | 1 | вед.инж. | 2 |
| Подбор и изучение научно-технической литературы | 1 | вед. инж. | 5 |
| Выбор инструментов разработки | 2 | вед. инж., инж.прогр.1 | 10 |
| Составление блок-схем программ | 1 | инж.прогр1. | 4 |
| Технический проект | Выяснение видов входных данных | 1 | инж.прогр2. | 2 |
| Разработка алгоритмов | 2 | вед. инж., инж.прогр1. | 15 |
| Разработка структуры программы | 2 | инж.прогр1, инж.прогр2. | 5 |
| Рабочий проект | Ввод исходных данных | 1 | инж.прогр.2 | 2 |
| Реализация выбранных алгоритмов | 1 | инж.прогр1, инж.прогр2. | 20 |
| Тестирование и устранение ошибок | 1 | инж.прогр.1 | 5 |
| Интеграция | Отладка | 1 | инж.прогр.2 | 5 |
| Написание документации | 1 | инж.прогр.2 | 15 |
| Сборка в составе программного комплекса | 1 | инж.прогр.1 | 5 |
|  | **Итого:** |  |  | 95 |

Для удобства реализации проводимых работ строится «дерево целей», которое разбивает поставленную задачу на ряд подзадач. «Дерево» разрабатывается на основе проведенного раньше распределения работ между участниками группы.

Рисунок 6.1. - Дерево целей



**6.3.Экономическая часть**

В экономической части проекта необходимо рассчитать смету затрат.

Смета рассчитывается по следующим статьям затрат:

1. Материалы, используемые на стадии выполнения ОКР/ОИР.
2. Заработная плата (основная и дополнительная) разработчиков изделия .
3. Страховые взносы (30% от основной и дополнительной заработной платы).
4. Прочие денежные расходы (25 % от суммы первых 4-х пунктов сметы затрат).

Таблица 6.3.Расчет заработной платы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Должность | Оклад, руб./мес. | Дневная зар. плата\*,  руб./день | Продолжительность работы каждого исполнителя, дней | Сумма,  руб.\*\* |
| Ведущий инженер  Инженер программист 1  Инженер программист 2 | 30 000  15 000  15 000 | 1364  682  682 | 32  64  49 | 43 648  43 648  33 418 |
| Итого: тарифная заработная плата | | | | 120 714 |
| Доплаты (премии, 50% от тарифной заработной платы) | | | | 60 357 |
| Основная заработная плата (тарифная з.п. + премии) | | | | 181 071 |
| Дополнительная заработная плата (20-25% от основной заработной платы) | | | | 36 214 |
| Основная и дополнительная заработная плата | | | | 217 285 |

\*Дневная заработная плата рассчитывается путем деления месячной заработной платы на 22 рабочих дня в месяц.

\*\* Величина тарифной з.п. каждого разработчика определяется умножением его дневной з.п. на количество отработанных им в данном проекте дней.

Полученные данные используются для составления сметы затрат.

Таблица 6.4. Смета затрат.

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование затрат | Сумма, тыс. руб. |
| 1.  2.  3.  4.  5. | Материалы (5 % от основной з.п.)  Основная заработная плата разработчиков  Дополнительная заработная плата  Страховы взносы (30 % от суммы основной и дополнительной з.п.)  Прочие расходы (25 % от суммы расходов по первым 4-м пунктам сметы затрат) | 9 054  181 071  36 214  65 186  72 881 |
|  | ИТОГО: | 364 406 |

6.4.Вывод

В экономической части дипломного проекта были выполнены следующие задачи:

1. Определена группа разработчиков в количестве 3 человек: ведущий инженер, 2 инженера программиста.
2. Рассчитано количество дней необходимое для решения поставленной задачи.
3. Построено дерево целей с указанием этапов проектирования и исполнителей.
4. Составлена смета затрат.

**7. Безопасность и экологичность проектных решений.**

**7.1. Цель и решаемые задачи**

При работе с ПЭВМ могут возникнуть потенциально опасные и вредные факторы, воздействие которых на организм человека может принести ему вред и привести к травматизму.

**7.2. Опасные и вредные факторы при работе с ПЭВМ**

Основные факторы с возможными последствиями изложены в ГОСТ 12.1.003-74/80 и сведены в таблицу 7.2.1.

Таблица 7.2.1. Основные опасные и вредные факторы.

|  |  |  |
| --- | --- | --- |
| № | Наименование фактора | Факторы, влияющие на человека |
| 1 | Повышенный уровень статического электричества | Электрический удар, пожар, взрыв |
| 2 | Повышенное значение напряжения электрической цепи | Электротравма |
| 3 | Электрическая дуга | Ожоги, пожар |
| 4 | Повышенная напряженность электрического поля и электромагнитного излучения | Нарушение ЦНС, и сердечно-сосудистой системы, утомляемость, раздражительность, раковые заболевания, нарушения психики, нарушения со стороны эндокринной системы, заболевания органов зрения. |
| 5 | Повышенная или пониженная температура воздуха, влажность, подвижность воздуха рабочей зоны | Перегрев или переохлаждение организма,простудные заболевания, обморожения, тепловой удар, заболевания кожи. |
| 6 | Неудовлетворительная освещенность рабочей зоны | Утомляемость, дискомфорт, опасность травматизма, ухудшение зрения |
| 7 | Повышенный уровень шума | Нервно-психическая перегрузка, заболевания органов слуха |
| 8 | Монотонность труда | Нервно-психическая перегрузка |
| 9 | Умственное перенапряжение | Изменение функционального состояния ЦНС |
| 10 | Повышение уровня химических веществ | Раздражающе действует на ЦНС, блокирование гемоглобина, нарушение тканевого дыхания, общетоксическое, канцерогенное действие на организм человека |

**7.3. Характеристика объекта исследования**

В данном разделе решаются вопросы безопасной жизнедеятельностипри работе с разрабакой компонента визуализации системы доставки расписания.

В офисе установлено 10 компьютеров оснащенных ЖК мониторами.

Системный блок типа: процессор – [INTEL Core i5 3470, LGA 1155, OEM](http://www.citilink.ru/catalog/parts/cpu/685395/) 3.2ГГц, материнская плата [ASUS P8B75-M LE LGA 1155, mATX, Ret](http://www.citilink.ru/catalog/parts/motherboards/678006/) чипсет Intel B75; память DDR3 — 2слота; частотой до 1600МГц, DVD-Rom, интегрированная звуковая карта. Корпус имеет матовую поверхность. Кнопки утоплены в корпус.

Так же в помещении установлено МФУ HP Photosmart Plus B210 series.

Все имеющиеся клавиатуры исполнены в виде отдельного устройства с возможностью свободного перемещения. На нижней части клавиатуры имеется опорное приспособление, позволяющее изменять угол наклона поверхности клавиатуры в пределах от 5 до 15 градусов.

Так же в помещении установлено МФУ HP Photosmart Plus B210 series.

**7.4. Мероприятия по безопасности труда и сохранению работоспособности.**

7.4.1. Обеспечение требований эргономики и технической эстетики

7.4.1.1. Планировка помещения и размещение оборудования

Помещения для ПЭВМ, в том числе помещения для работы с дисплеями, размещать в подвалах не допускается. Дверные проходы внутренних помещений вычислительных центров должны быть без порогов. При разных уровнях пола соседних помещений и местах перехода должны быть устроены наклонные плоскости (пандусы) с углом наклона не более 30 градусов.

На рисунке 7.4.1.1.1. показана существующая планировка помещения, в котором планируется использовать разрабатываемое программное обеспечение, и размещение оборудования.

Рисунок 7.4.1.1.1. Планировка помещения и размещение оборудования.

8000

5000



ПЭВМ устанавливаются и размещаются в соответствии с требованиями технических условий заводов-изготовителей. При размещении рабочих мест с ПЭВМ расстояние между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), должно быть не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов - не менее 1,2 м.

Влияние вредных электромагнитных излучений уменьшается за счет удаления их источников от оператора и установкой защитного экрана на монитор ПЭВМ.

Влияние загазованности, запыленности и вредных паров, выделяемых изоляцией установки, устраняется за счет правильного размещения оборудования, обеспечивающего хорошую естественную вентиляцию.

В кабинете имеется 1 окно, 9 столов, шкаф, кресла и компьютеры. Рассчитаем удельную площадь Пуд помещения, приходящиеся на одного человека по формуле:



где *Пп* - площадь помещения;

*По*  - площадь, занятая крупногабаритным оборудованием и мебелью;

 число человек, одновременно работающих в помещении.



В соответствии с СанПиН 2.2.2/2.4.1340-03, площадь на одно рабочее место пользователей ПЭВМ на базе электронно-лучевой трубки (ЭЛТ) должна составлять не менее 6 (при продолжительности работы менее 4 часов в день – 4,5), на базе жидкокристаллических мониторов - 4,5.

Удельная площадь кабинета *Пуд* = 4,5  на одного человека. Следовательно, нормативные требования СанПиН 2.2.2./2.4.1340-03 сходятся с расчетами. Эргономические решения и организация рабочего места пользователей ПЭВМ Требования к организации и оборудованию рабочего места пользователя ПЭВМ приведены в ГОСТ 12.2.032-78, СанПиН 2.2.2./2.4.1340-03.

Высота рабочей поверхности стола для пользователей должна регулироваться в пределах 680-800 мм; при отсутствии таковой возможности высота рабочей поверхности стола должна составлять 725 мм.

Модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать:

* ширину - 800, 1200, 1400 мм;
* глубину - 800 или 1000 мм;
* высоту (при нерегулируемой) - 725 мм.

Рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм, расстояние до нижнего края рабочей поверхности не менее 150 мм.

Рабочий стул (кресло) должен быть подъемно-поворотным с регулируемым по высоте и углам наклона сиденья и спинки, а также по расстоянию от спинки до переднего края сиденья:

* ширина и глубина сиденья не менее  400  *мм*  с закругленным краем и возможностью регулировки угла наклона вперед - до 15 градусов и назад - до 5 градусов;
* высота опорной поверхности спинки 300 ± 20 *мм*, ширина - не менее 380 *мм*, радиус кривизны - 400 *мм*;
* угол наклона спинки в вертикальной плоскости в пределах 0 ± 30 градусов;
* расстояние от спинки до переднего края сиденья должно регулироваться в пределах 260 - 400 *мм*.

Кресло должно иметь регулируемые стационарные или съемные подлокотники:

* длина не менее 250 мм и ширина – 50 -70 *мм*;
* высота над сиденьем -230 ± 30 *мм*;
* расстояние между подлокотниками – 350 - 500 *мм*.

Клавиатуру следует располагать на поверхности стола на расстоянии 100-300 *мм* от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

Конструкция применяемой клавиатуры выбиралась, исходя из следующих нормативных параметров:

* конструкция выполнена в виде отдельного устройства с возможностью свободного перемещения;
* опорное приспособление, позволяет изменять угол наклона поверхности клавиатуры в пределах от 5 до 15 градусов;
* высота среднего ряда клавиш 25 *мм*;
* заводское исполнение часто используемых клавиш выполнено в центре, внизу и справа, а редко используемых - вверху и слева;
* выделение цветом, размером, формой и местом расположения функциональных групп клавиш;
* оптимальный размер клавиш - 15 *мм*;
* клавиши с углублением в центре и шагом 19  ±  1 *мм*;
* расстояние между клавишами 3 *мм*;
* одинаковый ход всех клавиш с максимальным сопротивлением нажатию 1,5 *Н*;
* звуковая обратная связь включения клавиш с регулировкой уровня звукового сигнала. Возможностью ее отключения.

7.4.1.2. Цветовое оформление помещения

Решения, относящиеся к области технической эстетики, должны быть основаны на рекомендациях СН-181-70 по цветовому оформлению помещения. При выборе цветового оформления помещения необходимо учесть психофизиологическое влияние цвета на центральную нервную систему и орган зрения человека, оптико-физическое воздействие, основанное на отражающей способности цвета и эстетическое восприятие, обусловленное гармоничным сочетанием разных цветов.

При цветовом оформлении помещения необходимо учесть ориентацию окон в отношении сторон света и характер искусственного освещения. У данного помещения окна ориентированы на юг, стены – светло-желтые, а пол покрыт оранжевым паркетом. Потолок в помещении белого цвета. Параметры цветового оформления помещений приведены в таблице 7. 4.1.1.1.:

Таблица 7.4.1.1.1. Параметры цветового оформления помещений.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ориентация окон помещения | Наименование цвета (поверхности) | Характеристика цветов | | | N образца  CH 181-70 |
| длина волны, нм | чистота | коэффициент  отражения, % |
| Юг | Светло-желтый (стены) | 572 ± 5 | 31 ± 10 | 83 ± 7 | 5,5 |
| Оранжевый (пол) | 580 ± 7 | 38 ± 5 | 73 ± 7 | 4,4 |

Выбор образцов цвета для отделочных материалов и изделий следует осуществлять с учетом фактуры: поверхности в помещениях должны иметь матовую и полуматовую фактуру для исключения попадания отраженных бликов в глаза работающего.

7.4.2. Обеспечение оптимальных параметров воздуха рабочих зон

7.4.2.1. Нормирование параметров микроклимата

Принцип нормирования метеорологических условий производственной среды санитарными правилами и нормами СанПиН 2.2.4.548-96

Таблица 7.4.2.1.1. Оптимальные параметры микроклимата с ПЭВМ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Категория работ | Температура воздуха в °С не более | Относительная влажность воздуха, % | Скорость движения воздуха, *м/с* |
| Холодный | Легкая-1б | 21-23 | 40-60 | 0,1 |
| Теплый | Легкая-1б | 22-24 | 40-60 | 0,1 |

Примечание: К категории 1б относятся работы, производимые сидя, стоя или связанные с ходьбой и сопровождающиеся некоторым физическим напряжением, при которых расход энергии составляет от 140 до 174 *ккал/ч*.

В соответствии с СанПиН 2.2.4.548-96 работу операторов можно отнести к работе категории легкая-1б. Таким образом, фактические параметры микроклимата приведены в таблице 7.4.2.1.1.2.

Таблица 7.4.2.1.2. Фактические параметры микроклимата с ПЭВМ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Категория работ | Температура воздуха в °С | Относительная влажность воздуха, % | Скорость движения воздуха, *м/с* |
| Холодный | Легкая-1б | 22 | 50 | 0,1 |
| Теплый | Легкая-1б | 22 | 50 | 0,1 |

Из таблиц видно, что фактические параметры микроклимата в помещении соответствуют нормативным.

7.4.2.2. Нормирование уровней вредных химических веществ

Рассматривая загрязнение помещения вредными веществами внешней среды, необходимо, прежде всего, учитывать местоположение здания. Здание, в котором находится данное помещение, находится далеко от автострады. Вблизи нет промышленных предприятий. Наиболее частыми загрязнителями, попадающими из внешней среды в помещение, являются оксид углерода, диоксид азота, диоксид серы, пыль и др.

Здание состоит из бетонных конструкций, что служит источником поступления в помещение радона и торона. Из биоактивных соединений наиболее значимы: диоксид углерода, сероводород и др.

Мебель, одежда и обувь, хранимая в помещении, выделяют пыль с содержанием минерального волокна, углеводороды, полиэфирные смолы и другие соединения.

К наиболее опасным загрязнителям помещений относятся продукты курения, концентрация которых при большом количестве курящих людей в разное время рабочего дня в десятки раз выше, чем в их отсутствии. Поэтому необходимо организовать специально отведённые места для курения с вентиляцией этих зон. Возможный состав вредных веществ в данном помещении с указанием предельно допустимых концентраций отражен в таблице 7.4.2.2.1.

Таблица 7.4.2.2.1. Характеристика вредных веществ, содержащихся в воздухе помещения (ГОСТ 12.1.005-88 и ГН 2.2.5.1313-03).

|  |  |  |  |
| --- | --- | --- | --- |
| Вредные вещества | ПДК, *мг/м3* | Класс опасности | Действие на человека |
| 1. Внешние источники (от автострады) | | | |
| Оксид углерода | 20 | 4 | Блокирует гемоглобин, нарушает тканевое дыхание |
| Диоксид азота | 5 | 2 | Наркотическое действие, действие на кровеносную систему |
| Свинец (выхлопы автомобилей) | 0,01/0,0070 | 1 | Общетоксическое, канцерогенное |
| Пыль (сажа) | 4 | 4 | Раздражающее, канцерогенное |
| 2. Строительные материалы (бетонные конструкции) | | | |
| Радон, торон, полоний, уран | 0,015 | 1 | Канцерогенное, общетоксическое |
| 3. Мебель, одежда, обувь | | | |
| Фенопласты | 6 | 3 | Общетоксическое, аллергическое, канцерогенное |
| Полиэфирный лак | 6 | 2 | - |
| Капролактам | 10 | 3 | - |
| Формальдегид | 05 | 9 | - |
| Бензол | 5 | 2 | - |
| Пыль растительного и животного происхождения | 2-6 | 4 | - |
| 4. Антропоксины | | | |
| Диоксид углерода | 10 | 2 | Раздражающее, действует на ЦНС |
| Сероводород | 3 | 3 | - |
| Микробы |  |  | Общетоксическое |
| 5. Продукты курения | | | |
| Никотин | 10 | 3 | Наркотическое |

7.4.2.3. Нормирование уровней аэроионизации

Основное применение ионизаторов - создание в помещениях оптимальной концентрации отрицательно заряженных аэроионов(анионов), которые необходимы для нормальной жизнедеятельности. Лишенный аэроионов воздух - "мертвый", ухудшает здоровье и ведет к заболеваниям.

В таблице 6. приведем согласно СанПиН 2.2.4.1294-03 уровни положительных и отрицательных аэроионов в воздухе помещения:

Таблица 7.4.2.3.1. Уровни ионизации воздуха помещений при работе на ВДТ и ПЭВМ.

|  |  |  |
| --- | --- | --- |
| Уровни | Число ионов в 1 см. куб. воздуха | |
| n+ | n- |
| Минимально необходимые | 400 | 600 |
| Оптимальные | 1500-3000 | 3000-5000 |
| Максимально допустимые | 50000 | 50000 |

Аэроионы повышают умственную и физическую работоспособность, снимают стресс, укрепляют нервную систему, повышают сопротивляемость человеческого организма инфекционным заболеваниям. Катионы являются “отходами” дыхания, последствием электромагнитного излучения электроприборов. Увеличивая число отрицательно заряженных или легких ионов “-” заряда, можно самым благоприятным образом воздействовать на организм человека.

В помещении стоят компьютеры, есть кондиционеры. Первые в огромном количестве производят вредоносные положительные ионы, а вторые попросту “зачищают” воздух, превращая его в мертвый. Концентрация аэроионов снижается до 50-100 на 1см3.

Решить все эти проблемы можно с помощью искусственной аэроионизации(ионизаторов), а так же частым проветриванием помещений.

7.4.3 Создание рационального освещения

Работа за ПК относится к зрительным работам высокой точности для любого типа помещений. Искусственное освещение в помещениях эксплуатации ПК должно соответствовать действующим нормам освещения (СНиП 23-05-95). Помещение имеет естественное и искусственное освещение. В качестве источников света при искусственном освещении используются блоки офисных светильников ЛПО65-4х18 (4 лампы по 18 вт).

7.4.4 Защита от шума

Одним из наиболее распространенных факторов внешней среды, неблагоприятно воздействующих на организм человека, является шум. Источником являются механические устройства ЭВМ (принтеры и вентиляторы в блоках питания ПЭВМ). Нормирование шума производиться по ГОСТ 12.1.003-83 и СН 2.2.4/2.1.8.562-96.

Таблица 7.4.4.1. Нормируемые уровни звукового давления и звуки на рабочих местах (СН 2.2.4/2.1.8.562-96).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вид трудовой деятельности, рабочее место | Уровни звукового давления, дБ, в октавных полосах со среднегеометрическими частотами, Гц | | | | | | | | | Уровни звука в дБА |
|  | 31,5 | 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |  |
| Творческая деятельность, руко­во­дящая работа с повышенными тре­бо­ваниями, научная деятельность, конструирование и проектирование, программирование, преподавание и обучение, врачебная деятельность. Рабочие места в помещениях дирекции, проектно-конструк­торс­ких бюро, расчетчиков, програм­мис­тов вычислительных машин, в лабораториях для теоретических работ и обработки данных, приема больных в здравпунктах | 86 | 71 | 61 | 54 | 49 | 45 | 42 | 40 | 38 | 50 |

Фактическое значение уровня шума в офисе - системный блок компьютера, шум от качественно установленной системы вентиляции в офисном помещении составляет 45-47 дБА.

Уровень шума, поступающего извне можно снизить за счет уплотнения окон и двери. Так же можно оборудовать помещение звукопоглощающими поверхностями

7.4.5 Обеспечение режимов труда и отдыха

Режимы труда и отдыха при работе с ПЭВМ и ВДТ должны организовываться в зависимости от вида и категории трудовой деятельности. Есть три группы видов трудовой деятельности, в нашем случае это группа А - работа по считыванию информации с экрана ВДТ или ПЭВМ с предварительным запросом.

При выполнении в течение рабочей смены работ, относящихся к разным видам трудовой деятельности, за основную работу с ПЭВМ и ВДТ следует принимать такую, которая занимает не менее 50% времени в течение рабочей смены или рабочего дня.

Для видов трудовой деятельности устанавливается 3 категории тяжести и напряженности работы с ВДТ и ПЭВМ. В нашем случае для группы А - по суммарному числу считываемых знаков за рабочую смену, но не более 60 000 знаков за смену.

Для обозначения 3 категории труда, исходя из нашей группы А, укажем количество регламентированных перерывов, время их проведения и суммарное время на отдых:

Основным перерывом является перерыв на обед. В соответствии с особенностями трудовой деятельности пользователей ПЭВМ и характером функциональных изменений со стороны различных систем организма в режиме труда должны быть дополнительно введены два - три регламентированных перерыва длительностью 10 *мин*. каждый: два перерыва - при 8-часовом рабочем дне.

При 8-часовой смене с обеденным перерывом через 4 часа работы дополнительные регламентированные перерывы необходимо предоставлять через 3 часа работы и за 2 часа до ее окончания.

Режим труда и отдыха операторов ПЭВМ, непосредственно работающих с ВДТ, должен зависеть от характера выполняемой работы. При вводе данных, редактировании программ, чтении информации с экрана непрерывная продолжительность работы с ВДТ не должна превышать 4-х часов при 8 часовом рабочем дне,  через каждый час работы необходимо вводить перерыв на 5 - 10 *мин*., а через 2 часа - на 15 *мин*. Количество обрабатываемых символов (или знаков) на ВДТ не должно превышать 30 тыс. за 4 *ч.* работы.

В целях профилактики переутомления и перенапряжения при работе на ПЭВМ, в том числе при использовании дисплеев, необходимо выполнять во время регламентированных перерывов комплексы упражнений.

С целью снижения или устранения нервно-психического, зрительного и мышечного напряжения, предупреждения переутомления необходимо проводить сеансы психофизиологической разгрузки и снятия усталости во время регламентированных перерывов и после окончания рабочего дня.

Эти сеансы должны проводиться в специально оборудованном помещении - комнате психологической разгрузки. Эту комнату следует располагать на расстоянии не более 75 *м* от рабочих мест. Для снижения напряженности труда операторов ПЭВМ необходимо равномерно распределять их нагрузку и рационально чередовать характер деятельности - прием и выдачу результатов с работой за ПЭВМ и др.

7.4.6 Обеспечение электробезопасности

В соответствии с классификацией помещений по опасности поражения током (ПУЭ) помещение лаборатории относится к помещениям без повышенной опасности в виду отсутствия таких признаков как повышенная влажность (> 75 %), повышенная температура (T = + 35 °С), наличие токопроводящей пыли и полов.

Для защиты от поражения электрическим током в нормальном режиме должны быть применены по отдельности или в сочетании следующие меры защиты от прямого прикосновения:

основная изоляция токоведущих частей;

применение сверхнизкого (малого) напряжения.

Для дополнительной защиты от прямого прикосновения в электроустановках напряжением до 1 кВ, при наличии требований ПУЭ, следует применять устройства защитного отключения (УЗО).

Для защиты от поражения электрическим током в случае повреждения изоляции должны быть применены по отдельности или в сочетании следующие меры защиты при косвенном прикосновении:

* защитное заземление;
* автоматическое отключение питания;
* уравнивание потенциалов;
* выравнивание потенциалов;
* двойная или усиленная изоляция;
* сверхнизкое (малое) напряжение;
* защитное электрическое разделение цепей;
* изолирующие (непроводящие) помещения, зоны, площадки.

Проводники защитного заземления во всех электроустановках, а также нулевые защитные проводники в электроустановках напряжением до 1 кВ с глухо заземленной нейтралью, в том числе шины. Должны иметь буквенное обозначение РЕ и цветовое обозначение чередующимися продольными или поперечными полосами одинаковой ширины (для шин от 15 до 100 мм) желтого и зеленого цветов. Заземление имеет сопротивление равное 4 Ом, что соответствует требованиям ПУЭ.

Защиту при косвенном прикосновении следует выполнять во всех случаях, если напряжение в электроустановке превышает 50 В переменного и 120 В постоянного тока. На одной линии находятся по два ПЭВМ.

Устройство защитного отключения (УЗО), является в настоящее время одним из самых передовых способов защиты человека от поражения электрическим током, при косвенном прикосновении, обеспечиваемой путем автоматического отключения питания.

При малых токах замыкания, снижении уровня изоляции, а также при обрыве нулевого защитного проводника, УЗО является единственным средством защиты человека от электропоражения.

Другим не менее важным свойством УЗО является его способность осуществлять защиту от возгораний и пожаров, возникающих на объектах вследствие возможных повреждений изоляции, неисправностей электропроводки и электрооборудования.

УЗО, заблаговременно реагируя на ток утечки на землю, отключит электропроводку от источника питания, предупреждая тем самым недопустимый нагрев и последующее воспламенение.

В рассматриваемом помещении УЗО электромеханического типа, реагирующий на переменный синусоидальный и пульсирующий токи утечки (УЗО-Д типа А).

7.4.7 Защита от статического электричества

В течение работы на корпусе компьютера накапливается статическое электричество. На расстоянии 5-10 см от экрана напряженность электростатического поля составляет 60-280 кВ/м, то есть в 10 раз превышает норму 15 кВ/м. Для уменьшения напряжённости применяются увлажнители и нейтрализаторы, антистатическое покрытие пола. Заземление корпуса ЭВМ обеспечено подведением заземляющей жилы к питающим розеткам. Сопротивление заземления 4 Ом, согласно (ПУЭ) для электроустановок с напряжением до 1000 В.

7.4.8 Обеспечение допустимых уровней электромагнитных полей

Основным источником электромагнитных полей, связанных с охраной здоровья людей, использующих в своей работе персональные компьютеры, являются дисплеи (мониторы), особенно дисплеи с электронно-лучевыми трубками. Они представляют собой источники наиболее вредных излучений, неблагоприятно влияющих на здоровье оператора. Приведём извлечение из СанПиН 2.2.2/2.4.1340-03 в таблице 7.4.8.1.

Таблица 7.4.8.1. Временные допустимые уровни ЭМП, создаваемых ПЭВМ на рабочих местах.

|  |  |  |
| --- | --- | --- |
| Наименование параметров | | ВДУ |
| Напряженность  электрического поля | в диапазоне частот 5 Гц - 2 кГц | 25 В/м |
| в диапазоне частот 2 кГц - 400 кГц | 2,5 В/м |
| Плотность магнитного потока | в диапазоне частот 5 Гц - 2 кГц | 250 нТл |
| в диапазоне частот 2 кГц - 400 кГц | 25 нТл |
| Напряженность электростатического поля | | 15 кВ/м |
| Поверхностный электростатический потенциал экрана видеомонитора | | 500 В |

Для достижения ВДУ ЭМП предусмотрено применение следующих методов:

* экранирование рабочего места;
* удаление рабочего места от источника ЭМП;
* рациональное размещение в рабочем помещении оборудования, излучающего электромагнитную энергию.

7.4.9 Обеспечение пожарной безопасности

Для решения проблем пожаробезопасности нам необходимо сначала определить и обосновать категорию помещения, руководствуясь НПБ 105-03 извлечение в таблице 7.4.9.1.:

Таблица 7.4.9.1. Категории помещений по взрывопожарной и пожарной опасности.

|  |  |
| --- | --- |
| Категория помещения | Характеристика веществ и материалов, находящихся (обращающихся) в помещении |
| В1 — В4  пожароопасные | Горючие и трудногорючие жидкости, твердые горючие и трудногорючие вещества и материалы (в том числе пыли и волокна), способные при взаимодействии с водой, кислородом воздуха или друг с другом только гореть при условии, что помещения, в которых они имеются в наличии или обращаются, не относятся к категориям А или Б |

Одной из наиболее важных задач пожарной защиты является защита помещений от разрушений и обеспечение их достаточной прочности в условиях воздействия высоких температур при пожаре. Учитывая высокую стоимость электронного оборудования помещений, а также категорию его пожарной опасности, здание имеет 1 степень огнестойкости.

Таблица 7.4.9.2. Огнестойкость строительных конструкций(СНиП 21-01-97).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Степень огнестойкости здания | Предел огнестойкости строительных конструкций, не менее | | | | | | |
| Несущие элементы здания | Наружные ненесущие стены | Перекрытия междуэтажные  (в том числе чердачные и над подвалами) | Элементы бесчердачных покрытий | | Лестничные клетки | |
| Настилы  (в том числе с утеплителем) | Фермы, балки, прогоны | Внутренние стены | Марши и площадки лестниц |
| I | R 120 | RЕ З0 | REI 60 | RE 30 | R 30 | REI 120 | R 60 |
|  |  |  |  |  |  |  |  |

где R – потеря несущей способности;

E – потеря целостности;

I – потеря теплоизолирующей способности.

Цифрами обозначено время в минутах.

Для тушения пожаров на начальных стадиях широко применяются огнетушители.

В помещениях с ПЭВМ применяются главным образом углекислотные огнетушители, достоинством которых является высокая эффективность тушения пожара, сохранность электронного оборудования, диэлектрические свойства углекислого газа, что позволяет использовать эти огнетушители даже в том случае, когда не удается обесточить электроустановку сразу, при этих условиях возможный пожар можно отнести к категориям А и Е.

Все помещения необходимо оборудовать установками стационарного автоматического пожаротушения. Наиболее целесообразно применять установки газового тушения пожара, действие которых основано на быстром заполнении помещения огнетушащим газовым веществом с резким сжижением содержания в воздухе кислорода. При наличии стационарного автоматического пожаротушения, количество огнетушителей уменьшается в два раза.

Здания (сооружения, строения, пожарные отсеки и части зданий, сооружений, строений - помещения или группы помещений, функционально связанные между собой) подразделяются по классу функциональной пожарной опасности в зависимости от их назначения, а также от возраста, физического состояния и количества людей, находящихся в здании, сооружении, строении, возможности пребывания их в состоянии сна [15, гл.9, ст.32]. Cогласно СНиП 21-01-97 рассматриваемое помещение относится к классу функциональной безопасности - Ф4.3 (здания органов управления учреждений, проектно-конструкторских организаций, информационных и редакционно-издательских организаций, научных организаций, банков, контор, офисов)

7.4.9.1. Обеспечение безопасной эвакуации персонала

Эвакуационные выходы должны обеспечивать эвакуацию всех людей, находящихся в помещении в течение необходимого времени. Выходы считаются эвакуационными, если они ведут:

а) из помещений первого этажа непосредственно наружу или через вестибюль, коридор, лестничную клетку;

б) из помещений любого этажа, кроме первого, в коридоры, ведущие в лестничную клетку (в том числе через холл), при этом лестничные клетки должны иметь выход наружу непосредственно или через вестибюль, отделенный от примыкающих коридоров перегородкой с дверями;

в) из помещений в соседнее помещение на этом же этаже, обеспеченное выходами, указанными в подпунктах а) и б).

При наличии двух эвакуационных выходов из помещения и коридора они должны быть расположены рассредоточенно.

Определим минимальное расстояние L между двумя выходами из помещения снаружи определим по формуле:



Где р – периметр помещения, м;

n – число эвакуационных выходов.



Фактическое расстояние между эвакуационными выходами помещения равно 20м.

20 > 7,5 , следовательно, требование рассредоточенности эвакуационных выходов соблюдено.

Фактическое расстояние от наиболее удаленного рабочего места до эвакуационного выхода из помещения снаружи составляет 40 м, при этом объем помещения составляет 101 м3; категория помещения по пожароопасности – В; степень огнестойкости здания – І; плотность людского потока в общем проходе – 7 чел\м2. Как свидетельствуют данные, представленные в таблице 11, в этом случае наибольшее расстояние до эвакуационного выхода не должно превышать 40 м. Следовательно, требования СП 1.131 30.2009 и Федерального законодательства по обеспечению безопасности эвакуации людей при пожаре соблюдены.

Таблице 7.4.9.1.1. Наибольшие расстояния до эвакуационных выходов (СП 1.131 30.2009)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Объём помещения тыс.м3 | Категория помещения | Степень огнестойкости | Расстояние, м, при плотности людского потока в общем проходе, *чел/м2* | | |
| до 1 | от 1 до 3 | от 3 до 5 |
| До 15 | А и Б  В1-В3 | СО | 40 | 25 | 15 |
| I, II, III, IV, СО | 100 | 60 | 40 |
| III, IV, С1 | 70 | 40 | 30 |
| V, С2, С3 | 50 | 30 | 20 |
|  |  |  |  |  |  |

7.4.9.2. Средства извещения и сигнализации о пожаре

Выбор того или иного типа оборудования пожарной сигнализации производится с учетом множества факторов: климатических условий, конструктивных параметров объекта и т.д.

В здании находятся адресные дымовые пожарные извещатели, применение которых позволяет по адресу сработавшего пожарного извещателя определить место возгорания. Они установлены в каждом кабинете и коридоре. Эти извещатели связаны с пультом пожарной охраны. При задымлении раздается сигнал, который предупреждает сотрудников об опасности

7.4.9.3. Способы и средства тушения пожара

Таблица 7.4.9.3.1. Класс пожара и рекомендуемые огнетушащие средства (ППБ-01-03).

|  |  |  |
| --- | --- | --- |
| Класс пожара | Характеристика горючей среды или объекта | Огнетушащие средства |
| Е | Электроустановки, находящиеся под напряжением | Галоидоуглеводороды, диоксид углерода, порошки |
| А | Обычные твердые горючие материалы (дерево, уголь, бумага, резина, текстиль и др.) | Все виды огнетушащих средств (прежде всего – вода) |

7.4.9.4. Молниезащита объекта

Практически каждый производственный объект должен обеспечиваться молниезащитой. Основным нормативным документом для разработки комплекса средств молниезащиты является «Инструкция по устройству молниезащиты зданий, сооружений и промышленных коммуникаций» (СО 153-34.21.122-2003). В соответствии с нормами должна быть предусмотрена соответствующая молниезащита от прямых и непрямых ударов молнии. Уровень защиты зависит от важности объекта и возможных последствий при ударах молнии. Здание, где находится рассматрирваемое помещение, можно классифицировать как обычный объект. Обычные объекты - жилые и административные строения, а также здания и сооружения высотой не более 60 м, предназначенные для торговли, промышленного производства, сельского хозяйства.

Подсчитаем плотность ударов молнии в землю за год Ng в месте размещения объекта по формуле:

Ng = (6,7 \* Td)/100,

Где Td – средняя продолжительность гроз в часах, определенная по региональным картам интенсивности грозовой деятельности, Td = 1,5 ч.

Ng = (6,7 \* 1,5)/100 = 0,1 раз на 1км2/год.

Категория устройств молниезащиты III, так как нет взрывоопасных смесей. Комплекс средств молниезащиты зданий или сооружений включает в себя устройства защиты от прямых ударов молнии (внешняя молниезащитная система - МЗС) и устройства защиты от вторичных воздействий молнии (внутренняя МЗС). В частных случаях молниезащита может содержать только внешние или только внутренние устройства. В общем случае часть токов молнии протекает по элементам внутренней молниезащиты.

Рассматриваемое здание включает только внутренние устройства молниезащиты. Токи молнии, попадающие в молниеприемники, отводятся в заземлитель через систему токоотводов (спусков) и растекаются в земле.

Молниеприемником в данном случае служит алюминиевая кровля с толщиной металла не менее 7 мм (естественный молниеприемник). Естественным токоотводом является металлический каркас здания. В качестве заземляющих электродов используется подземная металлическая конструкция.

7.4.10. Мероприятия и средства по защите окружающей среды

Основные вредные факторы и их влияние на окружающую среду представлены в таблице 7.4.10.1.

Таблица 7.4.10.1. Экологически вредные факторы, которые расположены в непосредственной близости от исследуемого объекта, и их влияние на окружающую среду.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Экологически опасные объекты | Экологически значимые факторы | Ожидаемые экологические последствия |
| 1 | Автотранспорт | Выбросы химических веществ в атмосферу. | Вклад в развитие парникового эффекта и образование кислотных дождей, ухудшение здоровья населения и демографических показателей. |
| 2 | Транспортно-дорожные средства | Шум | Ухудшение здоровья населения и демографических показателей. |
| 3 | Высокоразвитая сеть подземных коммуникаций | Выбросы газов и жидкостей, находящихся под давлением. Нарушение несущей способности грунта. | Взрывы, пожары, оползни, провалы и т.д. |

**7.5. Расчет искусственного освещения**

Определим электрическую мощность осветительной установки *W*, количество светильников *N*, высоту подвеса светильников *Нр* и схему размещения светильников по потолку для создания общего равномерного освещения в помещении.

Длина помещения *А* = 8 *м*, ширина *В* = 5 *м*, высота *h* = 3 *м*.

Напряжение питающей сети *U* = 220 *В*. Расчет будем вести по методу светового потока, используя люминесцентные лампы.

Выбираем светильник типа TBS233 4xTL-D18W IC L, в котором применяются 4 лампы типа TL-D, т.е. люминесцентные, белого цвета, мощностью 18 Вт. Длина и ширина светильника 600 мм, расстояние от потолка до светильника 300 мм.

Минимальная освещенность для создания общего освещения определяется:



откуда необходимое количество светильников *N*, равно



где *Emin* – минимальная, нормируемая общая освещенность в помещении, *лк* (при использовании люминесцентных ламп *Emin* = 300 *лк*);

*Sn* – площадь пола в помещении, *Sn = A × B* = 6 × 3 = 18 *м2*;

*k* – коэффициент запаса (в помещении, где отсутствует выделение пыли *k*= 1,5);

*Fл*- световой поток, создаваемый одной лампой, *лм* (для лампы, используемой в данном помещении, *Fл* = 1100 лм);

*z* – коэффициент неравномерности освещения, *z* = 1,1 – 1,2;

- коэффициент использования светового потока.

Значение коэффициента  зависит от показателя помещения  и коэффициентов отражения стен *qст* и потолка *qпт*,а также от высоты подвеса светильников *Нр*.

Высота подвеса светильников *Нр* определяется как расстояние между уровнем горизонтальной рабочей поверхности *hраб* и светильником. ,, где *h* – высота помещения, *h* = 3 *м*; *hраб* – уровень (высота) рабочей поверхности, *hраб* = 0,8 *м;* *hпот* – расстояние между светильником и потолком, *hпот* = 0,3 *м*.  Определим показатель помещения :



Находим коэффициенты отражения стен и потолка:  *qcm* = 0,5; *qnm* = 0,7 =>  0,44; *n* – количество ламп в светильнике, *n* = 4 шт. Таким образом, количество светильников равно



Размещение светильников показано на рис. 7.4.11.1.

Рис. 7.5.1. Расположение осветительных установок.



**7.6. Расчет естественного освещения**

Определить необходимую площадь световых проемов (остекления) S0, число окон n, размещение окон и размеры окон. Помещение находится на 9 этаже здания. На противоположной стороне улицы дома отсутствуют. Длина помещения *А* = 8 *м*, ширина *В* = 5 *м*, высота *h* = 3 *м*.

Необходимая площадь окон для создания нормированной естественной освещенности помещения определяется по формуле:

*S0* = 

где *Sn* - площадь пола в кабинете;

 - минимальный (нормированный) коэффициент естественной освещенности;

 - коэффициент световой характеристики окна;

*R* - коэффициент, учитывающий затенение окна противостоящими зданиями;

*r0* - коэффициент светопропускания;

*r1* - коэффициент, учитывающий влияние отраженного света при боковом освещении.

*Sn* = *A* × *B* = 8 × 5 = 40 *м2*

Работа, которой занимаются операторы в кабинете, следует отнести к работам малой точности. =1 (при боковом освещении).

Необходимо определить параметры окна:

*h1* = *h0 + h1  - hраб*

Для этого зададим высоту окна *h0 =* 2 *м*, размещение по отношению к уровню рабочей поверхности *hраб* = 0,8 *м,* возвышение верхнего края окна над горизонтальной рабочей плоскостью *h1* = 1*м*. *h1* = 2 + 1 – 0,8 = 2,2 *м*

Необходимо определить еще два параметра:

отношение длины помещения *(A)* к его ширине *(В)*



отношение ширины помещения *(В)*  к его возвышению верхнего края окна над горизонтальной рабочей поверхностью *(h1)*



Исходя из расчетов,  = 13.

Так как противостоящие здания отсутствуют, то принимаем R = 1

Для окон со сдвоенными деревянными рамами в помещении категории Б (помещении без больших выделений пыли) и вертикальном расположении остекления *r0* = 0,4.

Величина *r1* зависит от средневзвешенного коэффициента отражения света от ограждающих поверхностей *qср*, который определяется по формуле:

Где qп ,qст ,qпт  - коэффициенты отражения от пола, стен и потолка. Sп ,Sст ,Sпт - площади пола, стен и потолка, соответственно.

qп = 0,3; qст = 0,5; qпт = 0,5.

Площадь пола стен и потолка равны:

*Sст* = *(А + В)* × 2 × *h* = (8 + 5) × 2 × 3 = 78 *м2*

*Sn = Snm =* 40 *м2*

Таким образом,

 = 

=> *r1* = 4

Определив все параметры, входящие в формулу по определению площади остекления *S0*, находим:



Площадь остекления должна быть равна не менее *Sо* = 3,25 *м2*. Зная высоту окна *h0 =* 2 *м* и ширину *b* = 2 *м*, вычислим площадь окна:



Определим необходимое количество окон:



Рис. 7.6.1. План помещения со схемой расположения окна и оборудования

8000



5000

**Литература**

**Книги**

1. Бланшет Ж. Саммерфилд М.

Qt 4: Программирование GUI на C++. – Пер. с англ. 2-е изд.,доп. –

М.: КУДИЦ-ПРЕСС, 2008. – 736с.;

2. Макконнелл С.

Совершенный код. Мастер-класс.: Пер. с англ. –

М.: Издательство «Русская редакция», 2010. – 896 с.

3. Мартин Р.

Чистый код: создание, анализ и рефакторинг. Библиотека программиста. –

СПб.: Питер, 2012 – 464 с.

5. Саммерфилд М.

Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на С++. –

Пер. с англ. – СПб.: Символ-Плюс, 2011. – 560 с., ил.

4. Шлее М.

Qt 4.5. Профессиональное программирование на C++. –

СПб.: БХВ – Петербург, 2010. – 896с.:ил.+DVD – (в подлиннике).

**Стандарты**

1. ГОСТ 12.1.005 – 88 «Система стандартов безопасности труда. Общие санитарно-гигиенические требования к воздуу рабочей зоны»;
2. ГОСТ 12.2.032 – 78 «Система стандартов безопасности труда. Рабочее место при выполнении работ сидя»;
3. СанПин 2.2.2/2.4.1340 – 03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»;
4. СанПин 2.2.4.548 – 96 «Гигиенические требования к микроклимату производственных помещений»;
5. СП 1.131 30.2009 «Система противопожарной защиты. Эвакуационные пути и выходы»;
6. СН 181-70 «Указания по проектированию цветовой отделки интерьеров производственных зданий промышленны предприятий»;
7. СН 2.2.4/2.1.8.562 – 96 «Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки»;
8. СНиП 21-01 – 97 «Пожарная безопасность зданий и сооружений»;
9. СНиП 23-05 – 95 «Естесственное и искусственное освещение».

**Интеренет-ресурсы**

1. С++

http://ru.wikipedia.org/wiki/C%2B%2B (Последнее изменение этой страницы: 11:40, 19 мая 2013.) – Рускоязычная википедия. Общая информация о языке С++;

1. Git

http://ru.wikipedia.org/wiki/Git (Последнее изменение этой страницы: 11:39, 18 мая 2013.) – Рускоязычная википедия. Общая информация о распределённой системе управления версиями файлов Git;

1. Mercurial

http://ru.wikipedia.org/wiki/Mercurial (Последнее изменение этой страницы: 05:55, 28 марта 2013.) – Рускоязычная википедия. Общая информация о системе управления версиями файлов Mercurial;

1. QT

http://ru.wikipedia.org/wiki/Qt (Последнее изменение этой страницы: 07:45, 14 мая 2013.) – Рускоязычная википедия. Общая информация о кросс-платформенном инструментарии разработки программного обеспечения на языке программирования C++;

1. SVN

http://ru.wikipedia.org/wiki/Subversion() – Рускоязычная википедия. Общая информация о системе управления версиями файлов SVN;

1. YB.ORM

http://code.google.com/p/yb-orm/wiki/About (Updated Jul 16, 2012) – Общая информация о проекте YB.ORM.

**Приложение 1. Листинг программы**

* 1. **Основная часть**

mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QFileDialog>

#include <QMessageBox>

#include <QProcess>

#include <qzipreader\_p.h>

#include <QFile>

#include <QMap>

#include <QTextStream>

#include <QString>

#include <QStringList>

#include "documentparser.h"

#include <QTableView>

#include "tablemodel.h"

#include <zlib.h>

#include <QByteArray>

#include "delegate.h"

#include "widget.h"

#include <QVBoxLayout>

#include <QPushButton>

#include <orm/MetaDataSingleton.h>

#include "domain/User.h"

#include "sharedobj.h"

#include "schedulemanager.h"

#define \_R QString::fromUtf8

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

QStringList getUsers();

~*MainWindow*();

private:

Ui::MainWindow \*ui;

public slots:

void setFilePath();

void parseFile();

void clearTable();

void saveTable();

};

#endif // MAINWINDOW\_H

mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

using namespace std;

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->setWindowTitle(\_R("Загрузка расписания"));

SharedObjSingleton::instance();

QObject::connect(ui->changeFileButton, SIGNAL(clicked()), this, SLOT(setFilePath()));

QObject::connect(ui->parseButton, SIGNAL(clicked()), this, SLOT(parseFile()));

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void

MainWindow::setFilePath()

{

QString filePath = QFileDialog::getOpenFileName(0, \_R("Файл расписания"), "", "\*.docx");

ui->filePathEdit->setText(filePath);

}

void

MainWindow::parseFile()

{

QString filePath = ui->filePathEdit->text();

if (filePath == "")

{

QMessageBox::information(0, \_R("Ошибка"), \_R("Файл не выбран"));

return;

}

QZipReader zip(filePath);

if (zip.exists())

{

try {

QByteArray content = zip.fileData("word/document.xml");

DocumentParser parser(content);

QVector<Row> rows = parser.getRows();

QTableView \*table = new QTableView();

TableModel \*model = new TableModel(rows);

table->*setModel*(model);

table->setItemDelegateForRow(0, new TableHeaderDelegate);

table->resizeColumnsToContents();

table->verticalHeader()->setResizeMode(QHeaderView::ResizeToContents);

table->horizontalHeader()->hide();

table->verticalHeader()->hide();

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

clearTable();

layout->addWidget(table);

QPushButton \*clearButton = new QPushButton();

clearButton->setObjectName("clearButton");

clearButton->setText(\_R("Очистить"));

QObject::connect(clearButton, SIGNAL(clicked()), this, SLOT(clearTable()));

QPushButton \*saveButton = new QPushButton();

saveButton->setObjectName("saveButton");

saveButton->setText(\_R("Сохранить"));

QObject::connect(saveButton, SIGNAL(clicked()), this, SLOT(saveTable()));

QStringList users = getUsers();

QComboBox \*cb = new QComboBox();

cb->addItems(users);

cb->setObjectName("userComboBox");

layout = (QVBoxLayout \*) ui->infoLayout;

layout->addWidget(cb);

layout = (QVBoxLayout \*) ui->buttonLayout;

layout->addWidget(saveButton);

layout->addWidget(clearButton);

}

catch(ParserEx)

{

QMessageBox::information(0, \_R("Ошибка"), \_R("Документ поврежден"));

}

}

zip.close();

}

void

MainWindow::saveTable()

{

QVBoxLayout \*layout = (QVBoxLayout \*) ui->infoLayout;

QComboBox \*cb = (QComboBox \*) layout->*takeAt*(0)->*widget*();

Yb::Session session(Yb::theMetaData::instance(), SharedObjSingleton::instance().getEngine());

qDebug() << "Name: " << cb->currentText();

Domain::User user = Yb::query<Domain::User>(session)

.filter\_by(Domain::User::c.name == cb->currentText()).one();

{

QStringList categories;

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

QTableView \*table = (QTableView \*) layout->*itemAt*(0)->*widget*();

QAbstractItemModel \*model = table->model();

for (int i = 0; i < model->*columnCount*(); ++i)

{

QModelIndex index = model->*index*(0, i, QModelIndex());

categories << index.data().toString();

}

SharedObjSingleton::instance().setCategories(categories);

}

ScheduleManager scheduleManager(&user);

scheduleManager.convertToScheduleObj();

QString msg = \_R("Расписание сохранено для пользователя: ");

msg += user.name.value();

QMessageBox::information(0, \_R("Радость"), msg);

}

void

MainWindow::clearTable()

{

QVBoxLayout \*layout = (QVBoxLayout \*) ui->tableLayout;

QLayoutItem \*item = layout->*takeAt*(0);

if (item)

{

QWidget \*widget = item->*widget*();

delete item;

layout->removeWidget(widget);

delete widget;

}

layout = (QVBoxLayout \*) ui->buttonLayout;

QLayoutItem \*clearButton = layout->*takeAt*(1);

if (clearButton)

{

QWidget \*widget = clearButton->*widget*();

delete clearButton;

layout->removeWidget(widget);

delete widget;

}

QLayoutItem \*saveButton = layout->*takeAt*(1);

if (saveButton)

{

QWidget \*widget = saveButton->*widget*();

delete saveButton;

layout->removeWidget(widget);

delete widget;

}

}

QStringList

MainWindow::getUsers()

{

QStringList userList;

Yb::Session session(Yb::theMetaData::instance(), SharedObjSingleton::instance().getEngine());

Yb::DomainResultSet<Domain::User> users = Yb::query<Domain::User>(session).all();

Yb::DomainResultSet<Domain::User>::iterator ibegin = users.begin(), iend = users.end();

for (; ibegin != iend; ++ibegin)

{

userList << ibegin->name.value();

}

return userList;

}

* 1. **Распознавание**

documentparser.h

#ifndef DOCUMENTPARSER\_H

#define DOCUMENTPARSER\_H

#include <QString>

#include <QDomDocument>

#include <QFile>

#include <stdexcept>

#include <string>

#include <QVector>

#include <QtDebug>

#include <QtGui>

class Cell{

public:

QVector<QString> blocks\_;

//Cell();

inline void addBlock(QString block){

blocks\_.push\_back(block);

}

inline int size(){

return blocks\_.size();

}

QString concatContent(QString separator = "\n"){

QString rezult = "";

foreach(QString text, blocks\_)

rezult += text + separator;

return rezult;

}

bool hasSpecialCharacter(){

foreach (QString content, blocks\_){

if (content.contains(":") || content.contains("-"))

return true;

}

return false;

}

};

class ParserEx: public std::runtime\_error{

public:

ParserEx(const std::string &ctx, const std::string &msg)

: std::runtime\_error(ctx + ": " + msg) {}

};

typedef QVector<Cell> Row;

class DocumentParser{

QDomNode table\_;

QDomNodeList rows\_;

QString parseBlock(QDomNode node);

public:

DocumentParser(const QByteArray &content, const QString &doctype = "docx");

QVector<Row> getRows();

};

#endif // DOCUMENTPARSER\_H

documentparser.cpp

#include "documentparser.h"

#include <QMessageBox>

using namespace std;

DocumentParser::DocumentParser(const QByteArray &content, const QString &doctype)

{

QDomDocument doc;

doc.setContent(content, false);

if (doctype == "docx")

{

QDomNodeList tables = doc.elementsByTagName("w:tbl");

if (tables.size() == 0)

throw ParserEx("DocumentParser", "no one table found in document");

table\_ = tables.at(0);

// tr обозначает строку

rows\_ = table\_.toElement().elementsByTagName("w:tr");

}

}

QVector<Row>

DocumentParser::getRows()

{

QVector<Row> rows\_vector;

for (int i = 0; i < rows\_.count(); ++i)

{

// tc - столбец

QDomNodeList cells = rows\_.at(i).toElement().elementsByTagName("w:tc");

Row row;

for (int j = 0; j < cells.count(); ++j)

{

// p - блок с записью (предположительно до перевода строки)

QDomNodeList blocks = cells.at(j).toElement().elementsByTagName("w:p");

Cell cell;

for (int k = 0; k < blocks.count(); ++k){

QString content = parseBlock(blocks.at(k));

cell.addBlock(content);

}

row.push\_back(cell);

}

rows\_vector.push\_back(row);

}

return rows\_vector;

}

QString

DocumentParser::parseBlock(QDomNode block)

{

QString textInBlock = "";

// В t текстовое содержимое

QDomNodeList contents = block.toElement().elementsByTagName("w:t");

for (int i = 0; i < contents.count(); ++i){

textInBlock += contents.at(i).toElement().text();

}

return textInBlock;

}

* 1. **Модель таблицы**

tablemodel.h

#ifndef TABLEMODEL\_H

#define TABLEMODEL\_H

#include <QAbstractTableModel>

#include <QModelIndex>

#include <vector>

#include <string>

#include "documentparser.h"

#include <QVector>

#include "classifier.h"

#include <QStringList>

#include <QComboBox>

#include "mainwindow.h"

#include <QtDebug>

#include "category.h"

class TableModel : public QAbstractTableModel{

Q\_OBJECT

private:

QVector< QVector<QString> > table\_;

public:

explicit TableModel(QVector<Row> table, QObject \*parent = 0);

int *rowCount*(const QModelIndex &parent = QModelIndex()) const{

Q\_UNUSED(parent);

return table\_.size();

}

int *columnCount*(const QModelIndex &parent = QModelIndex()) const{

Q\_UNUSED(parent);

return table\_.at(0).size();

}

Qt::ItemFlags *flags*(const QModelIndex &index) const{

Qt::ItemFlags theFlags = QAbstractTableModel::*flags*(index);

theFlags |= Qt::ItemIsSelectable|Qt::ItemIsEnabled;

if (index.isValid() && index.row() == 0)

theFlags |= Qt::ItemIsEditable;

return theFlags;

}

QVariant *data*(const QModelIndex &index, int role) const;

QVariant *headerData*(int section, Qt::Orientation orientation, int role) const;

bool *setData*(const QModelIndex &index, const QVariant &value, int role);

QVector<QString> getHead() const{

return table\_.at(0);

}

signals:

public slots:

};

#endif // TABLEMODEL\_H

tablemodel.cpp

#include "tablemodel.h"

using namespace std;

TableModel::TableModel(QVector<Row> table, QObject \*parent) :

QAbstractTableModel(parent){

table.remove(0);

SharedObjSingleton::instance().setRows(table);

CategoryFactory \*factory = new CategoryFactory();

AbstractCategory \*category = new DowCategory();

factory->addCategory(category);

category = new SubjectCategory();

factory->addCategory(category);

category = new DateCategory();

factory->addCategory(category);

category = new TimeCategory();

factory->addCategory(category);

category = new PlaceCategory();

factory->addCategory(category);

category = new CourseCategory();

factory->addCategory(category);

Classifier cf(table, factory);

table\_ = cf.getRows();

table\_.push\_front(cf.suggestionParse());

}

QVariant

TableModel::*data*(const QModelIndex &index, int role) const{

if (!index.isValid() || index.row() > table\_.size()

|| index.column() > table\_.at(0).size())

return QVariant::Invalid;

if (role == Qt::DisplayRole || role == Qt::EditRole){

QVector<QString> row = table\_.at(index.row());

return row.at(index.column());

}

return QVariant::Invalid;

}

QVariant

TableModel::*headerData*(int section, Qt::Orientation orientation, int role) const{

if (role != Qt::DisplayRole && role != Qt::EditRole

&& orientation != Qt::Vertical)

return QVariant();

return section + 1;

}

bool

TableModel::*setData*(const QModelIndex &index, const QVariant &value, int role){

if (index.row() == 0){

table\_[0][index.column()] = value.toString();

emit dataChanged(index, index);

return true;

}

return false;

}

* 1. **Классификация**

classifier.h

#ifndef CLASSIFIER\_H

#define CLASSIFIER\_H

#include "category.h"

#include "documentparser.h"

#include <QVector>

#include <QString>

#include <QStringList>

#include <QtDebug>

#include "mainwindow.h"

#include <QMultiMap>

#include <algorithm>

typedef struct ColumnStatisticTmp{

ColumnStatisticTmp():

words(0)

, symbols(0)

, numbers(0)

, maxStringSize(0)

, minStringSize(1000)

, hasComa(false)

, hasCodon(false)

, hasDash(false)

{}

void normalize(int count){

words = words/count;

symbols = symbols/count;

numbers = numbers/count;

}

double words;

double symbols;

double numbers;

int maxStringSize;

int minStringSize;

bool hasComa;

bool hasCodon;

bool hasDash;

} ColumnStatistic;

class CategoryFactory;

class Classifier{

CategoryFactory \*categories\_;

QVector<Row> source\_table\_;

QVector< QVector<QString> > rows\_;

void normalize(QVector<Row> &rows);

void calculateCharacteristic(const Cell &cell, ColumnStatistic &statistic);

QString analizeStatistic(ColumnStatistic statistic);

public:

static QStringList categories;

Classifier(QVector<Row> &rows, CategoryFactory \*categories);

QVector<QString> suggestionParse();

inline QVector< QVector<QString> > getRows() const{

return rows\_;

}

};

#endif // CLASSIFIER\_H

classifier.cpp

#include "classifier.h"

Classifier::Classifier(QVector<Row> &rows, CategoryFactory \*categories):

categories\_(categories)

, source\_table\_(rows){

normalize(rows);

}

void

Classifier::normalize(QVector<Row> &rows){

foreach (Row row, rows){

QVector<QString> new\_row;

foreach(Cell cell, row){

new\_row.push\_back(cell.concatContent());

}rows\_.push\_back(new\_row);

}

}

QVector<QString>

Classifier::suggestionParse(){

QVector<QString> headerList;

for (int i = 0; i < source\_table\_.at(0).size(); ++i){

QVector<Cell> currentColumn;

ColumnStatistic statistic;

for (int j = 0; j < source\_table\_.size(); ++j){

Cell currentCell = source\_table\_.at(j).at(i);

calculateCharacteristic(currentCell, statistic);

currentColumn.push\_back(currentCell);

}

statistic.normalize(source\_table\_.size());

QMultiMap<double, AbstractCategory\*> rez = categories\_->calculateProbability(statistic, currentColumn);

QString header = (rez.end()-1).value()->getName();

headerList.push\_back(header);

}return headerList;

}

void

Classifier::calculateCharacteristic(const Cell &cell, ColumnStatistic &statistic){

foreach (QString str, cell.blocks\_){

if (str.size() > statistic.maxStringSize)

statistic.maxStringSize = str.size();

if (str.size() < statistic.minStringSize)

statistic.minStringSize = str.size();

QStringList words = str.split(" ");

statistic.words += words.size();

foreach (QChar ch, str){

if (ch == QChar(','))

statistic.hasComa = true;

if (ch == QChar(':'))

statistic.hasCodon = true;

if (ch == QChar('-'))

statistic.hasDash = true;

if (ch.isDigit())

statistic.numbers++;

if (ch.isLetter())

statistic.symbols++;

}

}

}

* 1. **Определение категории**

category.h

#ifndef CATEGORY\_H

#define CATEGORY\_H

#include <QVector>

#include <QMultiMap>

#include <QString>

#include "classifier.h"

#include "documentparser.h"

#include "mainwindow.h"

#define \_R QString::fromUtf8

class AbstractCategory

{

protected:

int ruleCount\_;

QString name\_;

public:

virtual double *calculateProbability*(struct ColumnStatisticTmp&, QVector<Cell> &) =0;

QString getName() const

{

return name\_;

}

AbstractCategory(QString name, int ruleCount):

ruleCount\_(ruleCount)

, name\_(name)

{}

};

class DowCategory: public AbstractCategory

{

public:

DowCategory(): AbstractCategory(\_R("День"), 3)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class SubjectCategory: public AbstractCategory

{

public:

SubjectCategory(): AbstractCategory(\_R("Дисциплина"), 3)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class PlaceCategory: public AbstractCategory

{

public:

PlaceCategory(): AbstractCategory(\_R("Место"), 4)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class DateCategory: public AbstractCategory

{

public:

DateCategory(): AbstractCategory(\_R("Дата"), 3)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class TimeCategory: public AbstractCategory

{

public:

TimeCategory(): AbstractCategory(\_R("Время"), 3)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class CourseCategory: public AbstractCategory

{

public:

CourseCategory(): AbstractCategory(\_R("Курс"), 2)

{}

double *calculateProbability*(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

class CategoryFactory

{

QVector<AbstractCategory\*> category\_;

public:

CategoryFactory(){}

void addCategory(AbstractCategory \*category);

QMultiMap<double, AbstractCategory\*> calculateProbability(struct ColumnStatisticTmp &statistic,

QVector<Cell> &column);

};

#endif // CATEGORY\_H

category.cpp

#include "category.h"

double

DowCategory::*calculateProbability*(ColumnStatistic &statistic, QVector<Cell> &column)

{

double prob = 0;

if (statistic.words < 2.)

prob += 1;

if (statistic.maxStringSize < 12.)

prob += 1;

if (statistic.numbers == 0)

prob += 1;

return prob/ruleCount\_;

}

double

PlaceCategory::*calculateProbability*(ColumnStatisticTmp &statistic, QVector<Cell> &column)

{

double prob = 0;

if (statistic.symbols == 0.)

prob += 1;

if (statistic.words == 1.)

prob += 1;

if (statistic.hasComa)

prob += 1;

if (statistic.numbers == 4.)

prob += 1;

return prob/ruleCount\_;

}

double

SubjectCategory::*calculateProbability*(ColumnStatisticTmp &statistic, QVector<Cell> &column)

{

double prob = 0;

if (statistic.symbols > 2.)

prob += 1;

if (statistic.words > 1.)

prob += 1;

if (statistic.numbers < 1.)

prob += 1;

return prob/ruleCount\_;

}

double

CourseCategory::*calculateProbability*(ColumnStatisticTmp &statistic, QVector<Cell> &column)

{

double prob = 0;

if (statistic.words == 1.)

prob += 1;

if (statistic.symbols == 0.)

prob += 1;

return prob/ruleCount\_;

}

double

DateCategory::*calculateProbability*(ColumnStatisticTmp &statistic, QVector<Cell> &column)

{

double prob = 0;

if (statistic.symbols <= 1.)

prob += 1;

if (statistic.numbers > 0.)

prob += 1;

if (statistic.hasDash || statistic.hasCodon)

prob += 1;

foreach(Cell cell, column){

if (cell.hasSpecialCharacter()){

foreach(QString str, cell.blocks\_){

QStringList l = str.split("-");

if (l.count() > 1 && statistic.symbols <= 1.){

QStringList num = l[0].split(".");

if (num[1].toInt() > 12){

prob -= 1;

break;

}

}

}

}

}

return prob/ruleCount\_;

}

double

TimeCategory::*calculateProbability*(ColumnStatisticTmp &statistic, QVector<Cell> &column){

double prob = 0;

if (statistic.symbols <= 1.)

prob += 1;

if (statistic.numbers > 0.)

prob += 1;

if (statistic.hasDash || statistic.hasCodon)

prob += 1;

return prob/ruleCount\_;

}

void

CategoryFactory::addCategory(AbstractCategory \*category){

category\_.push\_back(category);

}

QMultiMap<double, AbstractCategory\*>

CategoryFactory::calculateProbability(ColumnStatistic &statistic, QVector<Cell> &column){

QMultiMap<double, AbstractCategory\*> probability;

foreach (AbstractCategory \*category, category\_)

{

double prob = category->*calculateProbability*(statistic, column);

probability.insert(prob, category);

}

return probability;

}

* 1. **Организация выпадающих списков**

delegate.h

#ifndef DELEGATE\_H

#define DELEGATE\_H

#include <QItemDelegate>

#include <QCombobox>

#include <QStringList>

#include "mainwindow.h"

#include <QtDebug>

class TableHeaderDelegate : public QItemDelegate{

public:

TableHeaderDelegate(QObject \*parent = 0);

QWidget\* *createEditor*(QWidget \*parent, const QStyleOptionViewItem &option,

const QModelIndex &index) const;

void *setEditorData*(QWidget \*editor, const QModelIndex &index) const;

void *setModelData*(QWidget \*editor, QAbstractItemModel \*model,

const QModelIndex &index) const;

};

#endif // DELEGATE\_H

delegate.cpp

#include "delegate.h"

#include <QLineEdit>

TableHeaderDelegate::TableHeaderDelegate(QObject \*parent){

}

QWidget\*

TableHeaderDelegate::*createEditor*(QWidget \*parent, const QStyleOptionViewItem &option,

const QModelIndex &index) const{

QComboBox \*combo\_box = new QComboBox(parent);

QStringList categories;

categories << \_R("Дата") << \_R("Время") << \_R("Место") << \_R("Дисциплина") << \_R("День") << \_R("Курс");

combo\_box->addItems(categories);

return combo\_box;

}

void

TableHeaderDelegate::*setEditorData*(QWidget \*editor, const QModelIndex &index) const{

QComboBox \*combo\_box = qobject\_cast<QComboBox \*>(editor);

combo\_box->setCurrentIndex(index.row());

}

void

TableHeaderDelegate::*setModelData*(QWidget \*editor, QAbstractItemModel \*model,

const QModelIndex &index) const{

QComboBox \*combo\_box = qobject\_cast<QComboBox \*>(editor);

QVariant val = combo\_box->currentText();

model->*setData*(index, val, Qt::EditRole);

}

* 1. **Преобразование данных и сохранение в базу данных**

schedulemanager.h

#ifndef SCHEDULEMANAGER\_H

#define SCHEDULEMANAGER\_H

#include <QVector>

#include "documentparser.h"

#include "mainwindow.h"

#include <QStringList>

#include <orm/MetaDataSingleton.h>

#include "domain/User.h"

#include "sharedobj.h"

#include <QMap>

#include "domain/Schedule.h"

#include "domain/Place.h"

#include <QHash>

#include "domain/Activity.h"

class ScheduleManager

{

typedef void(ScheduleManager::\*Handler)(int, int);

Yb::Session \*session\_;

Domain::User \*user\_;

QMap<QString, Handler> handlers\_;

QStringList categories\_;

QVector<Row> table\_;

bool specialCaseProcessed\_;

Domain::ScheduleHolder \*scheduleHolder\_;

void dowHandler(int row, int column);

void subjectHandler(int row, int column);

void placeHandler(int row, int column);

void timeHandler(int row, int column);

void dateHandler(int row, int column);

void courseHandler(int row, int column);

public:

ScheduleManager(Domain::User \*user);

void convertToScheduleObj(bool save = false);

~ScheduleManager()

{

delete session\_;

}

};

#endif // SCHEDULEMANAGER\_H

schedulemanager.cpp

#include "schedulemanager.h"

using namespace Domain;

using namespace Yb;

ScheduleManager::ScheduleManager(Domain::User \*user):

specialCaseProcessed\_(false)

{

user\_ = user;

session\_ = new Yb::Session(Yb::theMetaData::instance()

, SharedObjSingleton::instance().getEngine());

handlers\_.insert(\_R("День"), &ScheduleManager::dowHandler);

handlers\_.insert(\_R("Место"), &ScheduleManager::placeHandler);

handlers\_.insert(\_R("Курс"), &ScheduleManager::courseHandler);

handlers\_.insert(\_R("Дисциплина"), &ScheduleManager::subjectHandler);

handlers\_.insert(\_R("Дата"), &ScheduleManager::dateHandler);

handlers\_.insert(\_R("Время"), &ScheduleManager::timeHandler);

table\_ = SharedObjSingleton::instance().getRows();

categories\_ = SharedObjSingleton::instance().getCategories();

}

void

ScheduleManager::convertToScheduleObj(bool save)

{

for(int i = 0; i < table\_.size(); ++i)

{

specialCaseProcessed\_ = false;

scheduleHolder\_ = new ScheduleHolder(\*session\_);

for(int j = 0; j < table\_.at(0).size(); ++j)

{

Handler handler = handlers\_.value(categories\_.at(j));

(this->\*handler)(i, j);

}

if(save)

session\_->commit();

else

session\_->rollback();

//qDebug() << "Repeat: " << (\*scheduleHolder\_)->repeat\_type.value();

//qDebug() << "Date start: " << (\*scheduleHolder\_)->start\_dt.value();

//qDebug() << "Date end: " << (\*scheduleHolder\_)->end\_dt.value();

//qDebug() << "Place: " << (\*scheduleHolder\_)->place->name.value();

//qDebug() << "Time start: " << (\*scheduleHolder\_)->start\_time.value();

//qDebug() << "Time end: " << (\*scheduleHolder\_)->end\_time.value();

}

}

void

ScheduleManager::dowHandler(int row, int column)

{

Cell cell = table\_.at(row).at(column);

(\*scheduleHolder\_)->dow = cell.concatContent();

}

void

ScheduleManager::dateHandler(int row, int column)

{

Cell cell = table\_.at(row).at(column);

Cell subjCell = table\_.at(row)

.at(categories\_.indexOf(\_R("Дисциплина")));

if(subjCell.size() > 1)

return;

QStringList dates = cell.concatContent("").split("-");

(\*scheduleHolder\_)->start\_dt = QDateTime::fromString(dates[0], "dd.MM");

(\*scheduleHolder\_)->end\_dt = QDateTime::fromString(dates[1], "dd.MM");

}

void

ScheduleManager::subjectHandler(int row, int column)

{

Cell cell = table\_.at(row).at(column);

Cell timeCell = table\_.at(row).at(categories\_.indexOf(\_R("Время")));

Cell dateCell = table\_.at(row).at(categories\_.indexOf(\_R("Дата")));

if (cell.size() > 1 && !specialCaseProcessed\_)

{

QString subject\_name = cell.blocks\_.at(0);

for(int i = 1; i < cell.size(); ++i)

{

ScheduleHolder tmpSchedule(\*session\_);

SubjectHolder subject(\*session\_);

subject->name = subject\_name;

tmpSchedule->subject = subject;

tmpSchedule->repeat\_type = "once";

Cell tcell = table\_.at(row).at(categories\_.indexOf(\_R("День")));

tmpSchedule->dow = tcell.concatContent("");

PlaceHolder place(\*session\_);

tcell = table\_.at(row).at(categories\_.indexOf(\_R("Место")));

place->name = tcell.concatContent("");

place->short\_name = tcell.concatContent("");

tmpSchedule->place = place;

QString date = dateCell.blocks\_.at(i);

tmpSchedule->start\_dt = QDateTime::fromString(date, "dd.MM");

tmpSchedule->end\_dt = QDateTime::fromString(date, "dd.MM");

QString time = timeCell.concatContent("");

tmpSchedule->start\_time = time.toInt();

tmpSchedule->end\_time = time.toInt() + 5;

}

specialCaseProcessed\_ = true;

}

else

{

QString text = cell.concatContent("");

if(text.contains(\_R("лаб.")))

{

ActivityHolder activity(\*session\_);

activity->name = \_R("Лабораторые");

activity->short\_name = \_R("Лабораторые");

//(\*scheduleHolder\_)->activities.insert(\*activity);

}

if(text.contains(\_R("практ.")))

{

ActivityHolder activity(\*session\_);

activity->name = \_R("Практика");

activity->short\_name = \_R("Практика");

//(\*scheduleHolder\_)->activities.insert(\*activity);

}

if(text.contains(\_R("лекции")))

{

ActivityHolder activity(\*session\_);

activity->name = \_R("Лекция");

activity->short\_name = \_R("Лекция");

//(\*scheduleHolder\_)->activities.insert(\*activity);

}

if(text.contains(\_R("консультации")))

{

ActivityHolder activity(\*session\_);

activity->name = \_R("Консультация");

activity->short\_name = \_R("Консультация");

// (\*scheduleHolder\_)->activities.insert(\*activity);

}

SubjectHolder subject(\*session\_);

subject->name = text;

(\*scheduleHolder\_)->subject = subject;

(\*scheduleHolder\_)->repeat\_type = "each";

}

//qDebug() << "Subject name: " << (\*scheduleHolder\_)->subject->name.value();

}

void

ScheduleManager::timeHandler(int row, int column)

{

Cell cell = table\_.at(row).at(column);

Cell subjCell = table\_.at(row)

.at(categories\_.indexOf(\_R("Дисциплина")));

if(subjCell.size() > 1)

return;

if(cell.size() == 1)

{

QString time1 = cell.blocks\_.at(0);

QStringList times = time1.split("-");

(\*scheduleHolder\_)->start\_time = times[0].remove(".").toInt();

(\*scheduleHolder\_)->end\_time = times[1].remove(".").toInt();

}

else

{

QString time1 = cell.blocks\_.first();

QString time2 = cell.blocks\_.last();

QString time\_end = "";

if (time2.contains("-"))

time\_end = time2.split("-").at(1);

else

time\_end = time1.split("-").at(1);

QString time\_start = time1.split("-").at(0);

(\*scheduleHolder\_)->start\_time = time\_start.remove(".").toInt();

(\*scheduleHolder\_)->end\_time = time\_end.remove(".").toInt();

}

}

void

ScheduleManager::placeHandler(int row, int column)

{

Cell cell = table\_.at(row).at(column);

QString name = cell.concatContent("");

DomainResultSet<Place> rs = query<Place>(\*session\_)

.filter\_by(Place::c.name == name || Place::c.short\_name == name)

.all();

DomainResultSet<Place>::iterator it = rs.begin();

if (it == rs.end()) {

bool ok = true;

QString msg = \_R("В базе не найдена аудитория: ") + name

+ \_R("\nВыберете вариант");

QStringList places;

places << \_R("Добавить: ") + name;

QHash<QString, qlonglong> ids;

DomainResultSet<Place> rs = query<Place>(\*session\_).all();

DomainResultSet<Place>::iterator it = rs.begin();

for (; it != rs.end(); ++it) {

places << it->name.value();

ids.insert(it->name.value(), it->id.value());

}

QString place = QInputDialog::getItem(0, \_R("Выбор аудитории"),

msg,

places,

ok);

if (ok)

{

if (place.contains(\_R("Добавить")))

{

PlaceHolder placeHolder(\*session\_);

placeHolder->name = name;

placeHolder->short\_name = name;

(\*scheduleHolder\_)->place = placeHolder;

}

else

{

(\*scheduleHolder\_)->place = PlaceHolder(\*session\_, ids.value(place));

}

}

} else

(\*scheduleHolder\_)->place = PlaceHolder(\*session\_, it->id);

}

* 1. **Разделяемый ресурс (класс для хранения данных)**

sharedobj.h

#ifndef SHAREDOBJ\_H

#define SHAREDOBJ\_H

#include <orm/MetaDataSingleton.h>

#include <QStringList>

#include <QScopedPointer>

#include "documentparser.h"

typedef std::auto\_ptr<Yb::Engine> EnginePtr;

class SharedObj

{

private:

EnginePtr engine\_;

QStringList categories\_;

QVector<Row> rows\_;

public:

SharedObj();

void setCategories(QStringList categories);

void setRows(QVector<Row> rows);

inline Yb::Engine\* getEngine(){

return engine\_.get();

}

inline QStringList getCategories() const{

return categories\_;

}

inline QVector<Row> getRows() const{

return rows\_;

}

};

typedef Yb::SingletonHolder<SharedObj> SharedObjSingleton;

#endif // SHAREDOBJ\_H

sharedobj.cpp

#include "sharedobj.h"

SharedObj::SharedObj(){

Yb::init\_default\_meta();

std::auto\_ptr<Yb::SqlConnection> conn(new Yb::SqlConnection("mysql+qodbc3://schedule"));

engine\_.reset(new Yb::Engine(Yb::Engine::READ\_WRITE, conn));

engine\_->set\_echo(true);

}

void

SharedObj::setCategories(QStringList categories){

categories\_ = categories;

}

void

SharedObj::setRows(QVector<Row> rows){

rows\_ = rows;

}

* 1. **Форма**

mainwindow.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="enabled">

<bool>true</bool>

</property>

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>770</width>

<height>394</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralWidget">

<widget class="QWidget" name="horizontalLayoutWidget">

<property name="geometry">

<rect>

<x>10</x>

<y>50</y>

<width>239</width>

<height>25</height>

</rect>

</property>

<layout class="QHBoxLayout" name="buttonLayout">

<item>

<widget class="QPushButton" name="parseButton">

<property name="text">

<string>Распознать</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QWidget" name="verticalLayoutWidget">

<property name="geometry">

<rect>

<x>10</x>

<y>90</y>

<width>751</width>

<height>241</height>

</rect>

</property>

<layout class="QVBoxLayout" name="tableLayout">

<property name="leftMargin">

<number>0</number>

</property>

</layout>

</widget>

<widget class="QWidget" name="horizontalLayoutWidget\_2">

<property name="geometry">

<rect>

<x>9</x>

<y>9</y>

<width>222</width>

<height>25</height>

</rect>

</property>

<layout class="QHBoxLayout" name="horizontalLayout\_2">

<item>

<widget class="QLineEdit" name="filePathEdit"/>

</item>

<item>

<widget class="QPushButton" name="changeFileButton">

<property name="text">

<string>Выбрать файл</string>

</property>

</widget>

</item>

</layout>

</widget>

<widget class="QWidget" name="verticalLayoutWidget\_2">

<property name="geometry">

<rect>

<x>590</x>

<y>10</y>

<width>171</width>

<height>21</height>

</rect>

</property>

<layout class="QVBoxLayout" name="infoLayout"/>

</widget>

</widget>

<widget class="QMenuBar" name="menuBar">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>770</width>

<height>21</height>

</rect>

</property>

</widget>

<widget class="QToolBar" name="mainToolBar">

<attribute name="toolBarArea">

<enum>TopToolBarArea</enum>

</attribute>

<attribute name="toolBarBreak">

<bool>false</bool>

</attribute>

</widget>

<widget class="QStatusBar" name="statusBar"/>

</widget>

<layoutdefault spacing="6" margin="11"/>

<resources/>

<connections/>

</ui>

* 1. **Служебные классы YB.ORM**

User.h

#ifndef ORM\_DOMAIN\_\_USER\_\_INCLUDED

#define ORM\_DOMAIN\_\_USER\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

class LoginSession;

typedef Yb::DomainObjHolder<LoginSession> LoginSessionHolder;

class Schedule;

typedef Yb::DomainObjHolder<Schedule> ScheduleHolder;

// } AUTOGEN\_END

class User;

typedef Yb::DomainObjHolder<User> UserHolder;

class User: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, name, pass, email, login, status, facult, is\_superuser, phone;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_USER"); }

typedef Yb::DomainResultSet<User> ResultSet;

// static method 'find'

typedef std::vector<User> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

User(Yb::DomainObject \*owner, const Yb::String &prop\_name);

User();

User(const User &other);

explicit User(Yb::Session &session);

explicit User(Yb::DataObject::Ptr d);

User(Yb::Session &session, const Yb::Key &key);

User(Yb::Session &session, Yb::LongInt id);

User &operator=(const User &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::String, 1> name;

Yb::Property<Yb::String, 2> pass;

Yb::Property<Yb::String, 3> email;

Yb::Property<Yb::String, 4> login;

Yb::Property<Yb::LongInt, 5> status;

Yb::Property<Yb::LongInt, 6> facult;

Yb::Property<Yb::LongInt, 7> is\_superuser;

Yb::Property<Yb::String, 8> phone;

// relation properties

Yb::ManagedList<LoginSession> login\_sessions;

Yb::ManagedList<Schedule> classes;

Yb::ManagedList<Schedule> subscriptions;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

User.cpp

#include "domain/User.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

User::Columns User::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<User> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

User::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, name(\_T("NAME"), Yb::Value::STRING, 100, 0, Yb::Value(), \_T(""), \_T(""), \_T("name"), \_T("name"))

, pass(\_T("PASS"), Yb::Value::STRING, 100, 0, Yb::Value(), \_T(""), \_T(""), \_T("!"), \_T("pass"))

, email(\_T("EMAIL"), Yb::Value::STRING, 40, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("email"), \_T("email"))

, login(\_T("LOGIN"), Yb::Value::STRING, 40, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("login"), \_T("login"))

, status(\_T("STATUS"), Yb::Value::LONGINT, 0, 0, Yb::Value(\_T("1")), \_T(""), \_T(""), \_T("status"), \_T("status"))

, facult(\_T("FACULT"), Yb::Value::LONGINT, 0, 0, Yb::Value(\_T("1")), \_T(""), \_T(""), \_T("facult"), \_T("facult"))

, is\_superuser(\_T("IS\_SUPERUSER"), Yb::Value::LONGINT, 0, 0, Yb::Value(\_T("0")), \_T(""), \_T(""), \_T("!"), \_T("is\_superuser"))

, phone(\_T("PHONE"), Yb::Value::STRING, 40, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("phone"), \_T("phone"))

{}

void User::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << name << pass << email << login << status << facult << is\_superuser << phone;

}

void User::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_USER"), \_T("user"), \_T("User")));

t->set\_seq\_name(\_T("S\_USER"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void User::create\_relations\_meta(Yb::Relations &rels){

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("login\_sessions");

attr2[\_T("property")] = \_T("user");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("User"), attr1, \_T("LoginSession"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("key")] = \_T("TEACHER\_ID");

attr2[\_T("property")] = \_T("teacher");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("User"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("subscriptions");

attr2[\_T("key")] = \_T("RECEIVER\_ID");

attr2[\_T("property")] = \_T("receiver");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("User"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

User::User(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{}

User::User()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

set(5, Yb::Value((Yb::LongInt)1));

set(6, Yb::Value((Yb::LongInt)1));

set(7, Yb::Value((Yb::LongInt)0));

// } AUTOGEN\_END

}

User::User(const User &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{}

User::User(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_USER"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

set(5, Yb::Value((Yb::LongInt)1));

set(6, Yb::Value((Yb::LongInt)1));

set(7, Yb::Value((Yb::LongInt)0));

// } AUTOGEN\_END

save(session);

}

User::User(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{}

User::User(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{}

User::User(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_USER"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, pass(this)

, email(this)

, login(this)

, status(this)

, facult(this)

, is\_superuser(this)

, phone(this)

, login\_sessions(this, \_T("login\_sessions"))

, classes(this, \_T("classes"))

, subscriptions(this, \_T("subscriptions"))

// } AUTOGEN\_END

{}

User &User::operator=(const User &other){

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

User::ListPtr

User::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

User::ListPtr lst(new User::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_USER")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(User(\*it));

}

return lst;

}

struct UserRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_USER"),

Yb::CreatorPtr(new Yb::DomainCreator<User>()));

}

UserRegistrator() { register\_domain(); }

};

static UserRegistrator User\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

Schedule.h

#ifndef ORM\_DOMAIN\_\_SCHEDULE\_\_INCLUDED

#define ORM\_DOMAIN\_\_SCHEDULE\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

#include "domain/User.h"

#include "domain/User.h"

#include "domain/Subject.h"

#include "domain/Place.h"

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

class ScheduleActivity;

typedef Yb::DomainObjHolder<ScheduleActivity> ScheduleActivityHolder;

class ScheduleDate;

typedef Yb::DomainObjHolder<ScheduleDate> ScheduleDateHolder;

class ScheduleGroup;

typedef Yb::DomainObjHolder<ScheduleGroup> ScheduleGroupHolder;

// } AUTOGEN\_END

class Schedule;

typedef Yb::DomainObjHolder<Schedule> ScheduleHolder;

class Schedule: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, update\_dt, teacher\_id, receiver\_id, subject\_id, place\_id, start\_dt, end\_dt, dow, repeat\_type, start\_time, end\_time, sent, checked;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_SCHEDULE"); }

typedef Yb::DomainResultSet<Schedule> ResultSet;

// static method 'find'

typedef std::vector<Schedule> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

Schedule(Yb::DomainObject \*owner, const Yb::String &prop\_name);

Schedule();

Schedule(const Schedule &other);

explicit Schedule(Yb::Session &session);

explicit Schedule(Yb::DataObject::Ptr d);

Schedule(Yb::Session &session, const Yb::Key &key);

Schedule(Yb::Session &session, Yb::LongInt id);

Schedule &operator=(const Schedule &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::DateTime, 1> update\_dt;

Yb::Property<Yb::DateTime, 6> start\_dt;

Yb::Property<Yb::DateTime, 7> end\_dt;

Yb::Property<Yb::String, 8> dow;

Yb::Property<Yb::String, 9> repeat\_type;

Yb::Property<int, 10> start\_time;

Yb::Property<int, 11> end\_time;

Yb::Property<int, 12> sent;

Yb::Property<int, 13> checked;

// relation properties

UserHolder teacher;

UserHolder receiver;

PlaceHolder place;

SubjectHolder subject;

Yb::ManagedList<ScheduleDate> dates;

Yb::ManagedList<ScheduleActivity> activities;

Yb::ManagedList<ScheduleGroup> groups;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

Schedule.cpp

#include "domain/Schedule.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

Schedule::Columns Schedule::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<Schedule> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

Schedule::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, update\_dt(\_T("UPDATE\_DT"), Yb::Value::DATETIME, 0, 0, Yb::Value(\_T("sysdate")), \_T(""), \_T(""), \_T("update-dt"), \_T("update\_dt"))

, teacher\_id(\_T("TEACHER\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_USER"), \_T("ID"), \_T("teacher-id"), \_T("teacher\_id"))

, receiver\_id(\_T("RECEIVER\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_USER"), \_T("ID"), \_T("receiver-id"), \_T("receiver\_id"))

, subject\_id(\_T("SUBJECT\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_SUBJECT"), \_T("ID"), \_T("subject-id"), \_T("subject\_id"))

, place\_id(\_T("PLACE\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_PLACE"), \_T("ID"), \_T("place-id"), \_T("place\_id"))

, start\_dt(\_T("START\_DT"), Yb::Value::DATETIME, 0, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("start-dt"), \_T("start\_dt"))

, end\_dt(\_T("END\_DT"), Yb::Value::DATETIME, 0, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("end-dt"), \_T("end\_dt"))

, dow(\_T("DOW"), Yb::Value::STRING, 15, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("dow"), \_T("dow"))

, repeat\_type(\_T("REPEAT\_TYPE"), Yb::Value::STRING, 10, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("repeat-type"), \_T("repeat\_type"))

, start\_time(\_T("START\_TIME"), Yb::Value::INTEGER, 0, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("start-time"), \_T("start\_time"))

, end\_time(\_T("END\_TIME"), Yb::Value::INTEGER, 0, Yb::Column::NULLABLE, Yb::Value(), \_T(""), \_T(""), \_T("end-time"), \_T("end\_time"))

, sent(\_T("SENT"), Yb::Value::INTEGER, 0, 0, Yb::Value(\_T("0")), \_T(""), \_T(""), \_T("sent"), \_T("sent"))

, checked(\_T("CHECKED"), Yb::Value::INTEGER, 0, 0, Yb::Value(\_T("0")), \_T(""), \_T(""), \_T("checked"), \_T("checked"))

{}

void Schedule::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << update\_dt << teacher\_id << receiver\_id << subject\_id << place\_id << start\_dt << end\_dt << dow << repeat\_type << start\_time << end\_time << sent << checked;

}

void Schedule::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_SCHEDULE"), \_T("schedule"), \_T("Schedule")));

t->set\_seq\_name(\_T("S\_SCHEDULE"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void Schedule::create\_relations\_meta(Yb::Relations &rels)

{

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("key")] = \_T("TEACHER\_ID");

attr2[\_T("property")] = \_T("teacher");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("User"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("subscriptions");

attr2[\_T("key")] = \_T("RECEIVER\_ID");

attr2[\_T("property")] = \_T("receiver");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("User"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("property")] = \_T("place");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Place"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("property")] = \_T("subject");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Subject"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("dates");

attr2[\_T("property")] = \_T("schedule");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Schedule"), attr1, \_T("ScheduleDate"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("activities");

attr2[\_T("property")] = \_T("schedule");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Schedule"), attr1, \_T("ScheduleActivity"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("groups");

attr2[\_T("property")] = \_T("schedule");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Schedule"), attr1, \_T("ScheduleGroup"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

Schedule::Schedule(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{}

Schedule::Schedule()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

set(1, Yb::Value(Yb::now()));

set(12, Yb::Value((int)0));

set(13, Yb::Value((int)0));

// } AUTOGEN\_END

}

Schedule::Schedule(const Schedule &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{}

Schedule::Schedule(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_SCHEDULE"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

set(1, Yb::Value(Yb::now()));

set(12, Yb::Value((int)0));

set(13, Yb::Value((int)0));

// } AUTOGEN\_END

save(session);

}

Schedule::Schedule(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{}

Schedule::Schedule(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{}

Schedule::Schedule(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_SCHEDULE"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, update\_dt(this)

, start\_dt(this)

, end\_dt(this)

, dow(this)

, repeat\_type(this)

, start\_time(this)

, end\_time(this)

, sent(this)

, checked(this)

, teacher(this, \_T("teacher"))

, receiver(this, \_T("receiver"))

, place(this, \_T("place"))

, subject(this, \_T("subject"))

, dates(this, \_T("dates"))

, activities(this, \_T("activities"))

, groups(this, \_T("groups"))

// } AUTOGEN\_END

{}

Schedule &Schedule::operator=(const Schedule &other)

{

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

Schedule::ListPtr

Schedule::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

Schedule::ListPtr lst(new Schedule::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_SCHEDULE")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(Schedule(\*it));

}

return lst;

}

struct ScheduleRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_SCHEDULE"),

Yb::CreatorPtr(new Yb::DomainCreator<Schedule>()));

}

ScheduleRegistrator() { register\_domain(); }

};

static ScheduleRegistrator Schedule\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

Subject.h

#ifndef ORM\_DOMAIN\_\_SUBJECT\_\_INCLUDED

#define ORM\_DOMAIN\_\_SUBJECT\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

class Schedule;

typedef Yb::DomainObjHolder<Schedule> ScheduleHolder;

// } AUTOGEN\_END

class Subject;

typedef Yb::DomainObjHolder<Subject> SubjectHolder;

class Subject: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, name, short\_name;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_SUBJECT"); }

typedef Yb::DomainResultSet<Subject> ResultSet;

// static method 'find'

typedef std::vector<Subject> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors++i;

Subject(Yb::DomainObject \*owner, const Yb::String &prop\_name);

Subject();

Subject(const Subject &other);

explicit Subject(Yb::Session &session);

explicit Subject(Yb::DataObject::Ptr d);

Subject(Yb::Session &session, const Yb::Key &key);

Subject(Yb::Session &session, Yb::LongInt id);

Subject &operator=(const Subject &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::String, 1> name;

Yb::Property<Yb::String, 2> short\_name;

// relation properties

Yb::ManagedList<Schedule> classes;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

Subject.cpp

#include "domain/Subject.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

Subject::Columns Subject::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<Subject> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

Subject::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, name(\_T("NAME"), Yb::Value::STRING, 100, 0, Yb::Value(), \_T(""), \_T(""), \_T("name"), \_T("name"))

, short\_name(\_T("SHORT\_NAME"), Yb::Value::STRING, 20, 0, Yb::Value(), \_T(""), \_T(""), \_T("short-name"), \_T("short\_name"))

{}

void Subject::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << name << short\_name;

}

void Subject::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_SUBJECT"), \_T("subject"), \_T("Subject")));

t->set\_seq\_name(\_T("S\_SUBJECT"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void Subject::create\_relations\_meta(Yb::Relations &rels)

{

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("property")] = \_T("subject");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Subject"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

Subject::Subject(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Subject::Subject()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

}

Subject::Subject(const Subject &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Subject::Subject(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_SUBJECT"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

save(session);

}

Subject::Subject(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Subject::Subject(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Subject::Subject(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_SUBJECT"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Subject &Subject::operator=(const Subject &other)

{

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

Subject::ListPtr

Subject::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

Subject::ListPtr lst(new Subject::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_SUBJECT")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(Subject(\*it));

}

return lst;

}

struct SubjectRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_SUBJECT"),

Yb::CreatorPtr(new Yb::DomainCreator<Subject>()));

}

SubjectRegistrator() { register\_domain(); }

};

static SubjectRegistrator Subject\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

Place.h

#ifndef ORM\_DOMAIN\_\_PLACE\_\_INCLUDED

#define ORM\_DOMAIN\_\_PLACE\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

class Schedule;

typedef Yb::DomainObjHolder<Schedule> ScheduleHolder;

// } AUTOGEN\_END

class Place;

typedef Yb::DomainObjHolder<Place> PlaceHolder;

class Place: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, name, short\_name;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_PLACE"); }

typedef Yb::DomainResultSet<Place> ResultSet;

// static method 'find'

typedef std::vector<Place> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

Place(Yb::DomainObject \*owner, const Yb::String &prop\_name);

Place();

Place(const Place &other);

explicit Place(Yb::Session &session);

explicit Place(Yb::DataObject::Ptr d);

Place(Yb::Session &session, const Yb::Key &key);

Place(Yb::Session &session, Yb::LongInt id);

Place &operator=(const Place &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::String, 1> name;

Yb::Property<Yb::String, 2> short\_name;

// relation properties

Yb::ManagedList<Schedule> classes;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

Place.cpp

#include "domain/Place.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

Place::Columns Place::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<Place> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

Place::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, name(\_T("NAME"), Yb::Value::STRING, 100, 0, Yb::Value(), \_T(""), \_T(""), \_T("name"), \_T("name"))

, short\_name(\_T("SHORT\_NAME"), Yb::Value::STRING, 20, 0, Yb::Value(), \_T(""), \_T(""), \_T("short-name"), \_T("short\_name"))

{}

void Place::Columns::fill\_table(Yb::Table &tbl){

tbl << id << name << short\_name;

}

void Place::create\_tables\_meta(Yb::Tables &tbls){

Yb::Table::Ptr t(new Yb::Table(\_T("T\_PLACE"), \_T("place"), \_T("Place")));

t->set\_seq\_name(\_T("S\_PLACE"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void Place::create\_relations\_meta(Yb::Relations &rels){

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("classes");

attr2[\_T("property")] = \_T("place");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Place"), attr1, \_T("Schedule"), attr2, Yb::Relation::Restrict));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

Place::Place(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Place::Place()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

}

Place::Place(const Place &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Place::Place(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_PLACE"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

save(session);

}

Place::Place(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Place::Place(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Place::Place(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_PLACE"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

, classes(this, \_T("classes"))

// } AUTOGEN\_END

{}

Place &Place::operator=(const Place &other){

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

Place::ListPtr

Place::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by){

Place::ListPtr lst(new Place::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_PLACE")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(Place(\*it));

}

return lst;

}

struct PlaceRegistrator{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_PLACE"),

Yb::CreatorPtr(new Yb::DomainCreator<Place>()));

}

PlaceRegistrator() { register\_domain(); }

};

static PlaceRegistrator Place\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

Activity.h

#ifndef ORM\_DOMAIN\_\_ACTIVITY\_\_INCLUDED

#define ORM\_DOMAIN\_\_ACTIVITY\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

// } AUTOGEN\_END

class Activity;

typedef Yb::DomainObjHolder<Activity> ActivityHolder;

class Activity: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, name, short\_name;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_ACTIVITY"); }

typedef Yb::DomainResultSet<Activity> ResultSet;

// static method 'find'

typedef std::vector<Activity> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

Activity(Yb::DomainObject \*owner, const Yb::String &prop\_name);

Activity();

Activity(const Activity &other);

explicit Activity(Yb::Session &session);

explicit Activity(Yb::DataObject::Ptr d);

Activity(Yb::Session &session, const Yb::Key &key);

Activity(Yb::Session &session, Yb::LongInt id);

Activity &operator=(const Activity &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::String, 1> name;

Yb::Property<Yb::String, 2> short\_name;

// relation properties

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

Activity.cpp

#include "domain/Activity.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

Activity::Columns Activity::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<Activity> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

Activity::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, name(\_T("NAME"), Yb::Value::STRING, 100, 0, Yb::Value(), \_T(""), \_T(""), \_T("name"), \_T("name"))

, short\_name(\_T("SHORT\_NAME"), Yb::Value::STRING, 20, 0, Yb::Value(), \_T(""), \_T(""), \_T("short-name"), \_T("short\_name"))

{}

void Activity::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << name << short\_name;

}

void Activity::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_ACTIVITY"), \_T("activity"), \_T("Activity")));

t->set\_seq\_name(\_T("S\_ACTIVITY"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void Activity::create\_relations\_meta(Yb::Relations &rels)

{

{

Yb::Relation::AttrMap attr1, attr2;

attr2[\_T("property")] = \_T("activity");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Activity"), attr1, \_T("ScheduleActivity"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

Activity::Activity(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{}

Activity::Activity()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

}

Activity::Activity(const Activity &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{}

Activity::Activity(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_ACTIVITY"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

save(session);

}

Activity::Activity(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{}

Activity::Activity(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{}

Activity::Activity(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_ACTIVITY"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, name(this)

, short\_name(this)

// } AUTOGEN\_END

{}

Activity &Activity::operator=(const Activity &other)

{

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

Activity::ListPtr

Activity::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

Activity::ListPtr lst(new Activity::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_ACTIVITY")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(Activity(\*it));

}

return lst;

}

struct ActivityRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_ACTIVITY"),

Yb::CreatorPtr(new Yb::DomainCreator<Activity>()));

}

ActivityRegistrator() { register\_domain(); }

};

static ActivityRegistrator Activity\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

ScheduleActivity.h

#ifndef ORM\_DOMAIN\_\_SCHEDULEACTIVITY\_\_INCLUDED

#define ORM\_DOMAIN\_\_SCHEDULEACTIVITY\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

#include "domain/Schedule.h"

#include "domain/Activity.h"

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

// } AUTOGEN\_END

class ScheduleActivity;

typedef Yb::DomainObjHolder<ScheduleActivity> ScheduleActivityHolder;

class ScheduleActivity: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, schedule\_id, activity\_id;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_SCHEDULE\_ACTIVITY"); }

typedef Yb::DomainResultSet<ScheduleActivity> ResultSet;

// static method 'find'

typedef std::vector<ScheduleActivity> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

ScheduleActivity(Yb::DomainObject \*owner, const Yb::String &prop\_name);

ScheduleActivity();

ScheduleActivity(const ScheduleActivity &other);

explicit ScheduleActivity(Yb::Session &session);

explicit ScheduleActivity(Yb::DataObject::Ptr d);

ScheduleActivity(Yb::Session &session, const Yb::Key &key);

ScheduleActivity(Yb::Session &session, Yb::LongInt id);

ScheduleActivity &operator=(const ScheduleActivity &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

// relation properties

ScheduleHolder schedule;

ActivityHolder activity;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

ScheduleActivity.cpp

#include "domain/ScheduleActivity.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

ScheduleActivity::Columns ScheduleActivity::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<ScheduleActivity> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

ScheduleActivity::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, schedule\_id(\_T("SCHEDULE\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_SCHEDULE"), \_T("ID"), \_T("schedule-id"), \_T("schedule\_id"))

, activity\_id(\_T("ACTIVITY\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_ACTIVITY"), \_T("ID"), \_T("activity-id"), \_T("activity\_id"))

{}

void ScheduleActivity::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << schedule\_id << activity\_id;

}

void ScheduleActivity::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_SCHEDULE\_ACTIVITY"), \_T("schedule-activity"), \_T("ScheduleActivity")));

t->set\_seq\_name(\_T("S\_SCHEDULE\_ACTIVITY"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void ScheduleActivity::create\_relations\_meta(Yb::Relations &rels)

{

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("activities");

attr2[\_T("property")] = \_T("schedule");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Schedule"), attr1, \_T("ScheduleActivity"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

{

Yb::Relation::AttrMap attr1, attr2;

attr2[\_T("property")] = \_T("activity");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Activity"), attr1, \_T("ScheduleActivity"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

ScheduleActivity::ScheduleActivity(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{}

ScheduleActivity::ScheduleActivity()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

}

ScheduleActivity::ScheduleActivity(const ScheduleActivity &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{}

ScheduleActivity::ScheduleActivity(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_SCHEDULE\_ACTIVITY"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

save(session);

}

ScheduleActivity::ScheduleActivity(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{}

ScheduleActivity::ScheduleActivity(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{}

ScheduleActivity::ScheduleActivity(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_SCHEDULE\_ACTIVITY"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, schedule(this, \_T("schedule"))

, activity(this, \_T("activity"))

// } AUTOGEN\_END

{}

ScheduleActivity &ScheduleActivity::operator=(const ScheduleActivity &other)

{

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

ScheduleActivity::ListPtr

ScheduleActivity::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

ScheduleActivity::ListPtr lst(new ScheduleActivity::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_SCHEDULE\_ACTIVITY")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(ScheduleActivity(\*it));

}

return lst;

}

struct ScheduleActivityRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_SCHEDULE\_ACTIVITY"),

Yb::CreatorPtr(new Yb::DomainCreator<ScheduleActivity>()));

}

ScheduleActivityRegistrator() { register\_domain(); }

};

static ScheduleActivityRegistrator ScheduleActivity\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

ScheduleDate.h

#ifndef ORM\_DOMAIN\_\_SCHEDULEDATE\_\_INCLUDED

#define ORM\_DOMAIN\_\_SCHEDULEDATE\_\_INCLUDED

#include <orm/DomainObj.h>

// AUTOGEN\_BEGIN(001) {

#include "domain/Schedule.h"

// } AUTOGEN\_END

namespace Domain {

// AUTOGEN\_BEGIN(002) {

// } AUTOGEN\_END

class ScheduleDate;

typedef Yb::DomainObjHolder<ScheduleDate> ScheduleDateHolder;

class ScheduleDate: public Yb::DomainObject

{

public:

// AUTOGEN\_BEGIN(007) {

struct Columns {

Yb::Column id, item\_type, schedule\_id, dt;

Columns();

void fill\_table(Yb::Table &tbl);

};

static Columns c;

// } AUTOGEN\_END

static const Yb::String get\_table\_name() { return \_T("T\_SCHEDULE\_DATE"); }

typedef Yb::DomainResultSet<ScheduleDate> ResultSet;

// static method 'find'

typedef std::vector<ScheduleDate> List;

typedef std::auto\_ptr<List> ListPtr;

static ListPtr find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by = Yb::Expression());

// constructors

ScheduleDate(Yb::DomainObject \*owner, const Yb::String &prop\_name);

ScheduleDate();

ScheduleDate(const ScheduleDate &other);

explicit ScheduleDate(Yb::Session &session);

explicit ScheduleDate(Yb::DataObject::Ptr d);

ScheduleDate(Yb::Session &session, const Yb::Key &key);

ScheduleDate(Yb::Session &session, Yb::LongInt id);

ScheduleDate &operator=(const ScheduleDate &other);

static void create\_tables\_meta(Yb::Tables &tbls);

static void create\_relations\_meta(Yb::Relations &rels);

// AUTOGEN\_BEGIN(003) {

// properties

Yb::Property<Yb::LongInt, 0> id;

Yb::Property<Yb::String, 1> item\_type;

Yb::Property<Yb::DateTime, 3> dt;

// relation properties

ScheduleHolder schedule;

// } AUTOGEN\_END

};

} // namespace Domain

// vim:ts=4:sts=4:sw=4:et:

#endif

ScheduleDate.cpp

#include "domain/ScheduleDate.h"

#include <orm/DomainFactorySingleton.h>

namespace Domain {

ScheduleDate::Columns ScheduleDate::c;

namespace {

Yb::Tables tbls;

Yb::Relations rels;

Yb::DomainMetaDataCreator<ScheduleDate> mdc(tbls, rels);

}

// AUTOGEN\_BEGIN(008) {

ScheduleDate::Columns::Columns()

: id(\_T("ID"), Yb::Value::LONGINT, 0, Yb::Column::PK, Yb::Value(), \_T(""), \_T(""), \_T("id"), \_T("id"))

, item\_type(\_T("ITEM\_TYPE"), Yb::Value::STRING, 10, 0, Yb::Value(), \_T(""), \_T(""), \_T("item-type"), \_T("item\_type"))

, schedule\_id(\_T("SCHEDULE\_ID"), Yb::Value::LONGINT, 0, 0, Yb::Value(), \_T("T\_SCHEDULE"), \_T("ID"), \_T("schedule-id"), \_T("schedule\_id"))

, dt(\_T("DT"), Yb::Value::DATETIME, 0, 0, Yb::Value(), \_T(""), \_T(""), \_T("dt"), \_T("dt"))

{}

void ScheduleDate::Columns::fill\_table(Yb::Table &tbl)

{

tbl << id << item\_type << schedule\_id << dt;

}

void ScheduleDate::create\_tables\_meta(Yb::Tables &tbls)

{

Yb::Table::Ptr t(new Yb::Table(\_T("T\_SCHEDULE\_DATE"), \_T("schedule-date"), \_T("ScheduleDate")));

t->set\_seq\_name(\_T("S\_SCHEDULE\_DATE"));

c.fill\_table(\*t);

tbls.push\_back(t);

}

void ScheduleDate::create\_relations\_meta(Yb::Relations &rels)

{

{

Yb::Relation::AttrMap attr1, attr2;

attr1[\_T("property")] = \_T("dates");

attr2[\_T("property")] = \_T("schedule");

Yb::Relation::Ptr r(new Yb::Relation(Yb::Relation::ONE2MANY, \_T("Schedule"), attr1, \_T("ScheduleDate"), attr2, Yb::Relation::Delete));

rels.push\_back(r);

}

}

// } AUTOGEN\_END

ScheduleDate::ScheduleDate(Yb::DomainObject \*owner, const Yb::String &prop\_name)

: Yb::DomainObject(\*tbls[0], owner, prop\_name)

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{}

ScheduleDate::ScheduleDate()

: Yb::DomainObject(\*tbls[0])

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

}

ScheduleDate::ScheduleDate(const ScheduleDate &other)

: Yb::DomainObject(other)

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{}

ScheduleDate::ScheduleDate(Yb::Session &session)

: Yb::DomainObject(session.schema(), \_T("T\_SCHEDULE\_DATE"))

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{

// AUTOGEN\_BEGIN(006) {

// } AUTOGEN\_END

save(session);

}

ScheduleDate::ScheduleDate(Yb::DataObject::Ptr d)

: Yb::DomainObject(d)

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{}

ScheduleDate::ScheduleDate(Yb::Session &session, const Yb::Key &key)

: Yb::DomainObject(session, key)

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{}

ScheduleDate::ScheduleDate(Yb::Session &session, Yb::LongInt id)

: Yb::DomainObject(session, \_T("T\_SCHEDULE\_DATE"), id)

// AUTOGEN\_BEGIN(005) {

, id(this)

, item\_type(this)

, dt(this)

, schedule(this, \_T("schedule"))

// } AUTOGEN\_END

{}

ScheduleDate &ScheduleDate::operator=(const ScheduleDate &other)

{

if (this != &other) {

\*(Yb::DomainObject \*)this = (const Yb::DomainObject &)other;

}

return \*this;

}

ScheduleDate::ListPtr

ScheduleDate::find(Yb::Session &session,

const Yb::Expression &filter, const Yb::Expression &order\_by)

{

ScheduleDate::ListPtr lst(new ScheduleDate::List());

Yb::ObjectList rows;

session.load\_collection(rows, Yb::Expression(\_T("T\_SCHEDULE\_DATE")), filter, order\_by);

if (rows.size()) {

Yb::ObjectList::iterator it = rows.begin(), end = rows.end();

for (; it != end; ++it)

lst->push\_back(ScheduleDate(\*it));

}

return lst;

}

struct ScheduleDateRegistrator

{

static void register\_domain() {

Yb::theDomainFactory::instance().register\_creator(\_T("T\_SCHEDULE\_DATE"),

Yb::CreatorPtr(new Yb::DomainCreator<ScheduleDate>()));

}

ScheduleDateRegistrator() { register\_domain(); }

};

static ScheduleDateRegistrator ScheduleDate\_registrator;

} // end namespace Domain

// vim:ts=4:sts=4:sw=4:et:

**1.11. Схема базы данных**

<?xml version="1.0"?>

<schema>

<table name="T\_USER" class="User" sequence="S\_USER" xml-name="user">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="NAME" type="string" size="100" null="false" />

<column name="PASS" type="string" size="100" null="false" xml-name="!" />

<column name="EMAIL" type="string" size="40" />

<column name="LOGIN" type="string" size="40" />

<column name="STATUS" type="longint" null="false" default="1" />

<column name="FACULT" type="longint" null="false" default="1" />

<column name="IS\_SUPERUSER" type="longint" null="false" default="0" xml-name="!"/>

<column name="PHONE" type="string" size="40" />

</table>

<table name="T\_SESSION" class="LoginSession" sequence="S\_SESSION" xml-name="login-session">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="USR\_ID" type="longint" null="false">

<foreign-key table="T\_USER" />

</column>

<column name="APP\_NAME" type="string" size="100" null="false" />

<column name="BEGIN\_SESSION" type="datetime" default="sysdate" null="false" />

<column name="END\_SESSION" type="datetime" />

<column name="TOKEN" type="string" size="100" />

</table>

<table name="T\_SUBJECT" sequence="S\_SUBJECT" class="Subject" xml-name="subject">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="NAME" type="string" size="100" null="false" />

<column name="SHORT\_NAME" type="string" size="20" null="false" />

</table>

<table name="T\_PLACE" sequence="S\_PLACE" class="Place" xml-name="place">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="NAME" type="string" size="100" null="false" />

<column name="SHORT\_NAME" type="string" size="20" null="false" />

</table>

<table name="T\_ACTIVITY" sequence="S\_ACTIVITY" class="Activity" xml-name="activity">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="NAME" type="string" size="100" null="false" />

<column name="SHORT\_NAME" type="string" size="20" null="false" />

</table>

<table name="T\_STUD\_GROUP" sequence="S\_STUD\_GROUP" class="StudGroup" xml-name="stud-group">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="SPEC\_CODE" type="string" size='20' null="false" />

<column name="YEAR" type="integer" null="false" />

<column name="NAME\_CODE" type="string" size="20" null="false" />

<column name="FULLTIME" type="integer" null="false" />

<column name="SHORT" type="integer" null="false" />

<column name="GRAD\_TYPE" type="integer" null="false" />

</table>

<table name="T\_SCHEDULE" sequence="S\_SCHEDULE" class="Schedule" xml-name="schedule">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="UPDATE\_DT" type="datetime" null="false" default="sysdate" />

<column name="TEACHER\_ID" type="longint" null="false">

<foreign-key table="T\_USER" />

</column>

<column name="RECEIVER\_ID" type="longint" null="false">

<foreign-key table="T\_USER" />

</column>

<column name="SUBJECT\_ID" type="longint" null="false">

<foreign-key table="T\_SUBJECT" />

</column>

<column name="PLACE\_ID" type="longint" null="false">

<foreign-key table="T\_PLACE" />

</column>

<column name="START\_DT" type="datetime" />

<column name="END\_DT" type="datetime" />

<column name="DOW" type="string" size="15" />

<column name="REPEAT\_TYPE" type="string" size="10" />

<column name="START\_TIME" type="integer" />

<column name="END\_TIME" type="integer" />

<column name="SENT" type="integer" null="false" default="0" />

<column name="CHECKED" type="integer" null="false" default="0" />

</table>

<table name="T\_SCHEDULE\_DATE" sequence="S\_SCHEDULE\_DATE" class="ScheduleDate" xml-name="schedule-date">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="ITEM\_TYPE" type="string" size="10" null="false" />

<column name="SCHEDULE\_ID" type="longint" null="false">

<foreign-key table="T\_SCHEDULE" />

</column>

<column name="DT" type="datetime" null="false" />

</table>

<table name="T\_SCHEDULE\_GROUP" sequence="S\_SCHEDULE\_GROUP" class="ScheduleGroup" xml-name="schedule-group">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="SCHEDULE\_ID" type="longint" null="false">

<foreign-key table="T\_SCHEDULE" />

</column>

<column name="GROUP\_ID" type="longint" null="false">

<foreign-key table="T\_STUD\_GROUP" />

</column>

</table>

<table name="T\_SCHEDULE\_ACTIVITY" sequence="S\_SCHEDULE\_ACTIVITY" class="ScheduleActivity" xml-name="schedule-activity">

<column name="ID" type="longint">

<primary-key />

</column>

<column name="SCHEDULE\_ID" type="longint" null="false">

<foreign-key table="T\_SCHEDULE" />

</column>

<column name="ACTIVITY\_ID" type="longint" null="false">

<foreign-key table="T\_ACTIVITY" />

</column>

</table>

<relation type="one-to-many">

<one class="User" property="login\_sessions" />

<many class="LoginSession" property="user" />

</relation>

<relation type="one-to-many">

<one class="User" property="classes" />

<many class="Schedule" property="teacher" key="TEACHER\_ID" />

</relation>

<relation type="one-to-many">

<one class="User" property="subscriptions" />

<many class="Schedule" property="receiver" key="RECEIVER\_ID" />

</relation>

<relation type="one-to-many">

<one class="Place" property="classes" />

<many class="Schedule" property="place" />

</relation>

<relation type="one-to-many">

<one class="Subject" property="classes" />

<many class="Schedule" property="subject" />

</relation>

<relation type="one-to-many" cascade="delete">

<one class="Schedule" property="dates" />

<many class="ScheduleDate" property="schedule" />

</relation>

<relation type="one-to-many" cascade="delete">

<one class="Schedule" property="activities" />

<many class="ScheduleActivity" property="schedule" />

</relation>

<relation type="one-to-many" cascade="delete">

<one class="Activity" />

<many class="ScheduleActivity" property="activity" />

</relation>

<relation type="one-to-many" cascade="delete">

<one class="Schedule" property="groups" />

<many class="ScheduleGroup" property="schedule" />

</relation>

<relation type="one-to-many" cascade="delete">

<one class="StudGroup" />

<many class="ScheduleGroup" property="group" />

</relation>

</schema>

**Приложение 2. Руководство пользователя**

Назначение

Программа предназначена для ускорения ввода данных в систему доставки расписания занятий. Программа представляет собой модуль для импорта данных с визуальным интерфейсом.

**Установка**

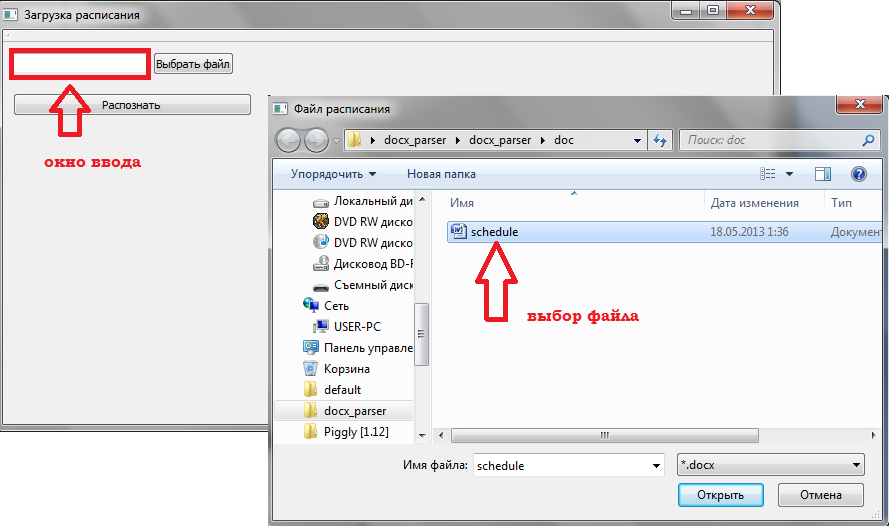
Программный модуль устанавливается вместе с программой для редактирования расписании системы доставки расписания занятий. Для работы необходимы: драйвер ODBC, Windows XP или выше, Linux, Mac OS, 512Mб RAM.

**Выполнение программы**

Для импорта данных подходят документы с расширением \*.odt, \*.docx. В файле обязательно должна быть таблица с расписанием, которое следует импортировать.

Для импортирования файла необходимо выбрать его, нажав на кнопку «Выбрать файл» и выбрав нужный. Можно так же самостоятельно указать путь к файлу в окно ввода.

Рисунок A 2.1.



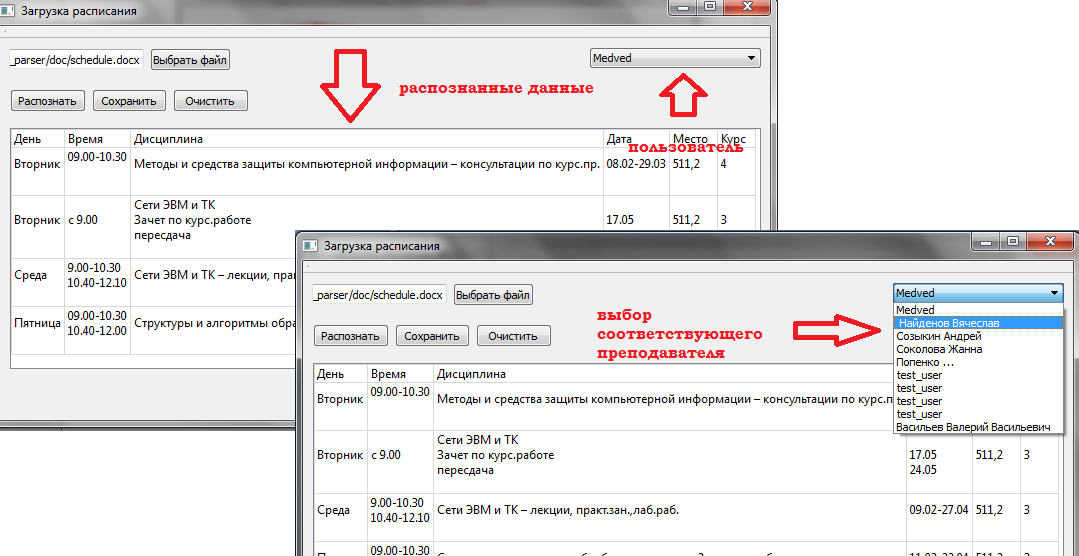
После того как выбор сделан необходимо нажать на кнопку «Распознать» для того, чтобы программа могла распознать поступившую информацию. Если файл был введен не тот, который нужно, его всегда можно изменить, нажав на кнопку «Выбрать файл» и снова « Распознать».

Рисунок A 2.2.



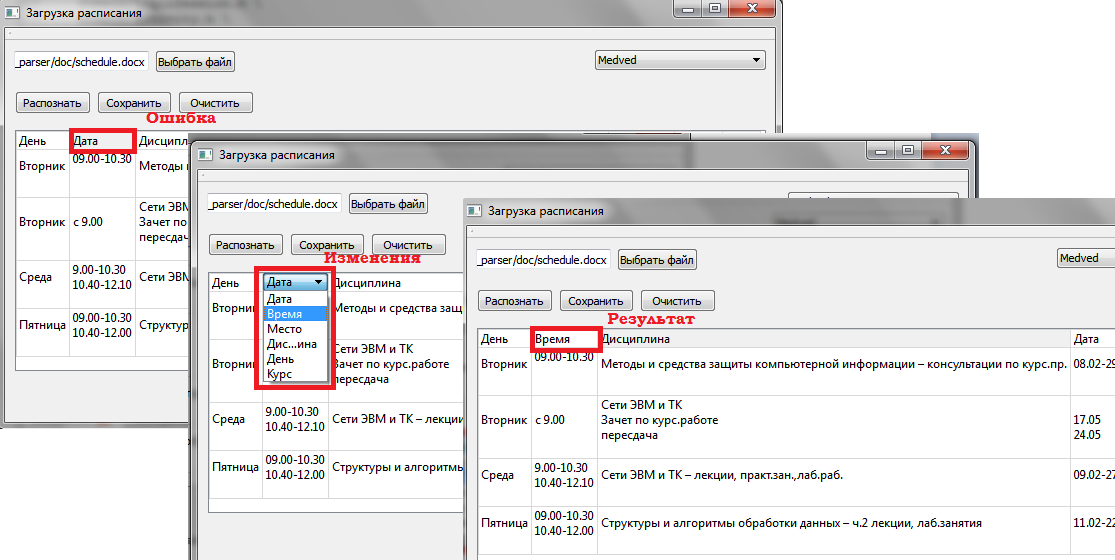
Распознав введенную информацию, программа выдаст на форме полученные данные в виде таблицы. Для занесения данных необходимо определить преподавателя, для которого введены данные.

Рисунок А 2.3.



Прежде чем сохранить, необходимо просмотреть полученную информацию и в случае ошибок внести корректировки.

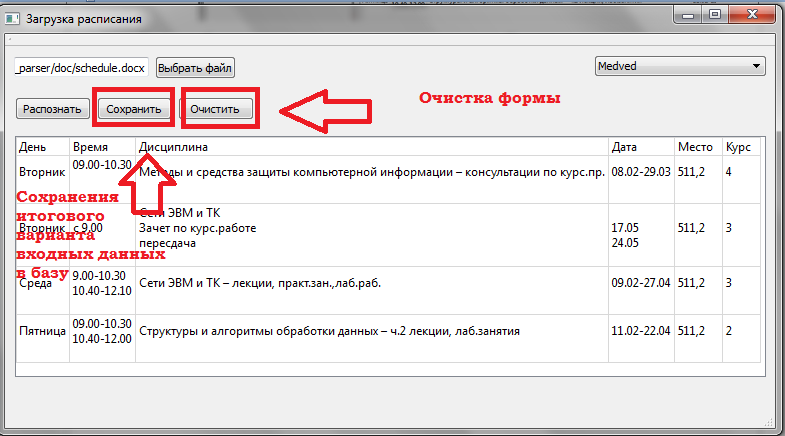
Рисунок А 2.4.



Если таблица соответствует ожиданием, необходимо нажать «сохранить» для занесения данных в базу. Возможно, потребуется уточнение данных связанное с особенностями обозначения информации в базе.

Так же есть возможность очистить форму и начать все заново.

Рисунок А 2.5.



**Сообщения оператору**

В ходе работы с модулем программа будет выдавать сообщения. Это могут быть как сообщения об ошибках, так и сообщения об успехе операции.

Возможные варианты сообщений:

1. Сообщения об ошибках могут возникать:
   * + если не указать файл, а сразу нажать «Распознать».
     + если подать на вход документ без таблицы.
2. Сообщения об успехе выполнения. Например для операции сохранения.
3. Сообщения-уточнения для выбора нужного варианта.

Рисунок A 2.6.

