Программа для составления расписания кафедры

Н. Д. Запольская
СПБГЭТУ «ЛЭТИ»
zapolskaya@mail.ru

А. В. Захарьин
СПБГЭТУ «ЛЭТИ»
blong03@mail.ru

Ю. В. Павлова
СПБГЭТУ «ЛЭТИ»
ivpavlova@etu.ru

***Аннотация*. В мире всё больше и больше людей переходят на цифровой документооборот, университеты не исключение, большинство бумаг, в том числе расписание, составляется в электронном виде. В силу того, что прогресс, а иногда обстоятельства, меняются быстрее чем люди успевают привыкнуть, научится работать с новой средой. Программа, созданная нами, и была призвана решить одно из проявлений этой проблемы.**

Ключевые слова: цифровой документооборот, электронное расписание, автоматизация, программное обеспечение, база данных, интерфейс, оптимизация, Windows 10, управление данными, конфликты расписания

I. Вступление

Первым пунктом необходимо сказать, что описанная ниже программа была создана с учётом запросов конкретных людей, под конкретные задачи определённой кафедры, структура программы может быть оптимизирована на других кафедрах с некоторыми изменениями базы данных и интерфейса, но только в рамках одного университета. Попытка внедрения в стороннем учебном заведении повлечёт большое количества переделок. Так же хочется отметить, что первоначальной задачей было решение конкретной проблемы, создание программы для широкого использования не предполагалось, вторичной задачей была возможность создания на основе получившегося продукта инструменты для работы на смежных кафедрах ВУЗа, но не более.

Первоначальная проблема заключалась в том, что электронное расписание, приходящее на почту старшему преподавателю, им же вручную обрабатывалось и составлялось согласно пожеланию остальных коллег. А это означает – огромная бумажная-ручная работа, неудобство ввода и большая затрата времени на ввод одной и той же информации.

Краткое описание первоначальных задач, поставленных перед нами, и которые мы должны были решить:

1) Ускорить работу по созданию расписания, за счет автоматизации процесса ввода.

2) Программа должна работать на офисных ПК.

3) Программа должна работать с операционной системой Windows 10.

II. Этап проектирования

Описания данного этапа начнётся с небольшого пролога, который позволил нам выбрать правильный вектор развития проекта. Наша команда не первая, кто пытался реализовать данную задумку, к сожалению, все предшественники потерпели неудачу, поэтому изучение вопроса началось с изучения неудач. Основной проблемой, которая по результатам была выявлена, это не возможность описания алгоритма выставления преподавателей в расписание при конфликтных ситуациях, на кафедре не существовало единой договорённости по этому поводу, люди просто договаривались между собой, программе же эти договорённости не объяснить. Исходя из соображения что программа должна учитывать человеческий фактор было принято решение оставить расстановку расписания на человеке. А вот весь остальной процесс автоматизировать. (То есть, в первую очередь, нам необходимо ускорить и упростить создание расписания).

III. Этап создания

В этой части будет описано как на практике воплощалась идея, озвученная в предыдущем этапе.

Основой для программы должна была послужить база данных, в которой бы хранилась вся необходимая информация о преподавателях, группе, временных промежутках, видах пар. Так же необходим был интерфейс, в котором бы пользователь с помощью простых форм и сортировок составлял бы расписание.

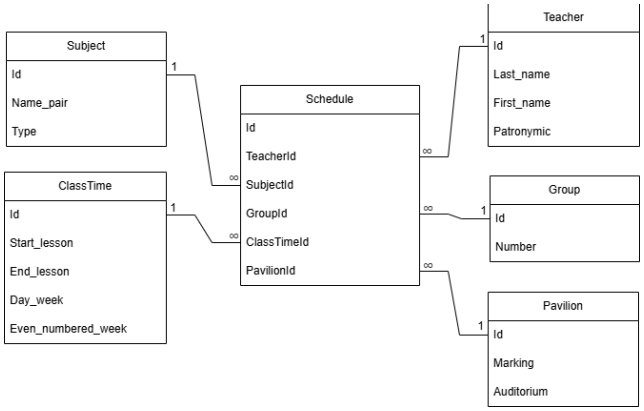

Рис. 1. Схема базы данных

Опишем тело программы, которое выполняет роль связи между базой данных и интерфейсом на уровне ПК, а также осуществляет загрузку и выгрузку информации.

На рис. 1 представлена архитектура базы данных, главная таблица, в которой содержится вся информация и к которой обращается программа. Остальные таблицы зависимые, в них содержится подробная информация, представленная в полях главной таблицы. Содержащаяся в них информация записана по строчкам и представлена в полях главной таблицы в виде ключей. Кроме того, зависимые таблицы связаны с программой и их можно редактировать во вкладках интерфейса.

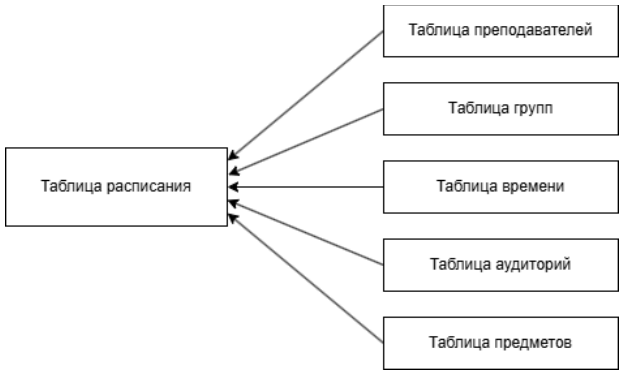

Рис. 2. Схема связи окон интерфейса

Интерфейс схематично выглядит как отдельные окна информации. Каждое окно отвечает за свою информацию (преподаватели, группы, аудитории, пары, расписание). При изменении любой из записей в любом окне происходит запись изменений в БД. Связь прямая и линейная, что обеспечивает высокую стабильность работы программы, минимизируя возникновения непредусмотренных исключений (т. е. запись в БД происходит сразу после того, как пользователь закончил редактировать запись).

A. Проблемы, с которыми столкнулись

1) Изучение языка и правильности его написания, поскольку написания программы для пользователя, в том числе интерфейс пользователя было для команды новым опытом, пришлось изучать информацию и экспериментировать с разными конструкциями и вариациями написания программного кода.

2) Оптимизация программы для пользователя, одной из основных задач была работоспособность программы без постоянной программной поддержки.

3) Аппаратные ограничения, программа обязана была запускаться на «офисном» ПК.

4) Получившийся продукт должен был быть эластичен, для дальнейшей его оптимизации на других кафедрах.

B. Схема программы

Схему работы программы можно разбить на несколько разделов. Каждый из разделов отвечает за свою область программы и содержит некоторое количество классов (рис. 3).

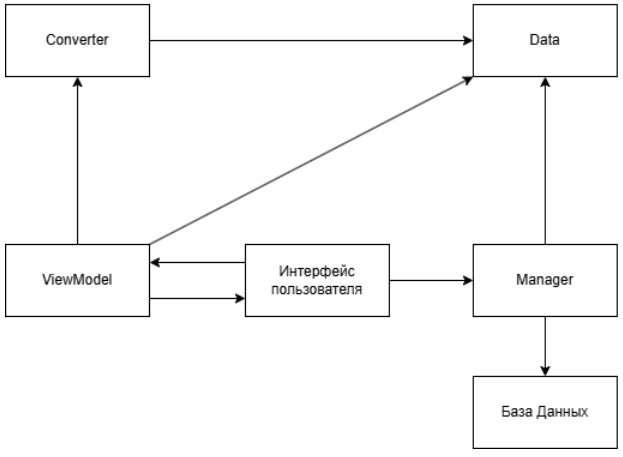

Рис. 3. Схема работы программы

Рассмотрим каждый из разделов подробнее:

1) Раздел «Data» – содержит 6 классов отвечающие за хранение информации, которая находиться в БД (для более быстрого доступа к ней).

2) Раздел «Converter» – содержит классы, которые переводят информацию из удобной для хранения в ту, которую удобно читать пользователю. Именно они отвечают за перевод, например, из цифр 1-7, которые находятся в базе, в читаемые «Понедельник», «Вторник», … , «Воскресенье».

Благодаря этим классам можно легко и быстро изменить отображение данных по усмотрению пользователя (Например: заменить «Понедельник» на «пон.»).

3) Раздел «Manager» – содержит классы для работы с базой данных. А именно чтение, запись, обновление информации, удаление и т. д. Это реализовано 7 классами: одним основным, который отвечает за подключение и 6-ю побочными (наследуемыми от основного), которые выполняют непосредственно чтение данных из каждой таблицы (Каждый класс отвечает за свою таблицу).

4) Раздел «ViewModel» – содержит классы, которые работают с формами интерфейса пользователя, служат для связи интерфейса с программой (своеобразный переводчик информации и команд из интерфейса в программные операции). Состоит также из 6 классов, каждый из которых создан под каждую вкладку («Расписание», «Преподаватели» и т. д.). Поэтому в один момент может работать только 1 класс, а именно тот, чья действующая вкладка сейчас активна. Классы устроены так, что они отлавливают любое изменение, которое происходит в разделах таблицы и, если надо, реагируют на него.

В отличии от кнопок, которые находятся на форме «добавить», «обновить», «удалить» (функционал которых отчасти прописан в самом классе формы) данные классы выполняют более детальную проработку.

Для большего понимания приведем пример: Пользователь заходит во вкладку преподаватели и хочет найти какого-то преподавателя. Для поиска по фамилии есть область (вверху), пользователь вводит туда фамилию и видит изменения в самой таблице. Тут и происходит работа одного из этих классов, а именно того, который отвечает за эту таблицу. Что же происходит? Класс зарегистрировал изменения в окне для ввода, понял, что пользователь захотел отфильтровать информацию и произвел сортировку по совпадениям, скрывая все неподходящие записи.

Тут вдруг пользователь замечает, что фамилия, которую он искал записана неправильно. Он наводит курсор мыши на соответствующую ячейку и нажимает. Тут снова срабатывает наш класс и видит, что пользователь изменил действующую ячейку, а это значит, что надо изменить и поля, которые находятся права от таблицы, а значит надо отправить изменения на форму, что он и делает.

Подводя итог можно сказать, что классы «view model» занимаются динамикой, реализуют столь привычные вещи, присутствие которых мы и не замечаем, а вот отсутствие данного класса может сделать программу неработоспособной.

IV. Заключение

На данный момент программа доведена до состояние полной работоспособности, преподаватели кафедры Физики в данный момент пользуются и изучают её, результат полностью оправдал ожидания. Есть два момента которые будут ещё доделаны, но данные функции – это наблюдение, которое будет направлено на увеличение удобства пользования.

Список литературы

[1] Иванов А.В. Цифровой документооборот в образовательных учреждениях. Москва: Изд-во МГУ, 2020. 256 с.

[2] Петров В.Н., Сидоров Е.А. Автоматизация процессов в университетах: проблемы и решения. Санкт-Петербург: Наука и Техника, 2019. 312 с.

[3] Смирнов Д.И. Программное обеспечение для управления данными в ВУЗах. Новосибирск: Изд-во НГУ, 2021. 224 с.

[4] Кузнецов А.П. Интерфейсы пользователя для образовательных программ. Екатеринбург: Уральский университет, 2018. 198 с.

[5] Федоров М.В., Захаров И.Л. Оптимизация работы кафедры с помощью цифровых технологий. Казань: Изд-во КФУ, 2022. 275 с.

[6] Беляев С.А. Управление расписанием учебных занятий в условиях конфликта. Ростов-на-Дону: Изд-во ЮФУ, 2020. 190 с.

[7] Григорьев Р.В. Программирование на C#: создание приложений для Windows 10. Москва: Бином, 2017. 300 с.

[8] Николаев Ю.К. Базы данных для образовательных учреждений. Томск: Изд-во ТГУ, 2019. 210 с.