АКИМОВА Д. А.

РАЗРАБОТКА ЭЛЕКТРОННОГО ПЛАНИРОВЩИКА ПРОЦЕССА ОБУЧЕНИЯ

Аннотация. Статья посвящена описанию разработки программного обеспечения, предлагающего пользователю помощь в проектировании образовательной программы. Визуализируя вводимые данные в двух вариантах – ориентированный граф и списки по семестрам, приложение облегчает задачу планировщику.

Ключевые слова: образовательная программа, планирование последовательности обучения, учебный план, визуальное представление информации, Microsoft Visual Studio 2022, Windows Presentation Foundation, .NET Core.

AKIMOVA D. A.

DEVELOPMENT OF ELECTRONIC PLANNER FOR DESIGING THE CURRICULUM

Abstract. The article presents a description of the development of the software to assist in designing the curriculum. The application facilitates the work of the planner by visualizing the data in two variants, oriented graph and lists by semester.

Keywords: training program, planning a sequence of the education, curriculum, visual presentation of information, Microsoft Visual Studio 2022, Windows Presentation Foundation, .NET Core.

Качество подготовки специалиста во многом определяется программой его обучения и учебным планом вуза. Задача последнего состоит в том, чтобы верно распределить нагрузку на студента на протяжении всего обучения, при этом сохраняя корректную последовательность изучаемых предметов и связей между ними. Здесь на помощь приходит программа, помогающая визуализировать ее.

Приложения из подобной направленности, выполняющие в той или иной степени вышеуказанное требование, уже существуют. Например, Шахтинская программа [1], обладающая большой базой данных, которую заполняет пользователь на основе уже заранее разработанного образовательного плана. Программное обеспечение (ПО) имеет обширный функционал, однако не является хорошим помощником в создании программы обучения учащихся. Описанное ниже приложение никак не конфликтует с Шахтинской программой, поскольку и выполняют они разные задачи. Также стоит упомянуть программу Obsidian [2], представляющую собой хранилище данных, предлагающую широкий набор действий по манипулированию данными. Это ПО и подобные ей решают общие задачи, но не узко направленные.

Суть программы состоит в том, чтобы провести человека через весь путь проектирования учебного плана, начиная с его «чернового» варианта, где известны лишь взаимосвязи нескольких дисциплин, а представление о полной очередности всех позиций еще не сформировано (т.е. визуализация происходит в виде ориентированного графа), и заканчивая уже готовым табличным видом, т.е. списком дисциплин, упорядоченным по семестру и алфавиту. При том, программа была направлена на пользователей ОС Windows, поскольку это самая распространенная и популярная операционная система на портативный компьютер на момент создания приложения.

Касательно функционала, программа должна была упрощать построение учебного плана, также обеспечивая систему проверок во избежание ошибок системы. Разрабатываемое ПО должно было позволять, как добавление дисциплины, так и ее удаление, а также создание связей между уже зарегистрированными позициями. К тому же существовала необходимость в редактировании уже созданной дисциплины и возможность меняться между режимом ориентированного графа и табличным видом, где столбцами выступают семестры. Программа должна была быть интуитивно понятна пользователю, легко передаваема через различные сервисы (Google Drive, Yandex Disk) или запоминающие устройства (USB-флеш-накопитель) и реализована в средах, позволяющих запуск программы на ОС Windows актуальных версий.

Любая программа имеет различные условия своего создания и представление того, как она должна выглядеть. Вышеописанные требования были сформулированы и написаны с помощью одного из международных стандартов ISO/IEC TR 19759:2005 [3].

Для реализации программы с учетом всех требований был выбран язык программирования C#. Так как этот язык изначально был предназначен для разработчиков на Windows, то платформа .NET тесно связана с этой операционной системой. Работа осуществлялась в интегрированной среде разработки (IDE) Microsoft Visual Studio 2022 с применением графической подсистемы Windows Presentation Foundation (WPF) в составе .NET Framework, использующей язык XAML. Эти инструменты для реализации программы также выбирались исходя из критерия относительно простой установки без скачивания сторонних программ. Все, что возможно потребуется для поддержки ПО, система предлагает скачать сама с прямыми ссылками.

С самого начала стоял вопрос о способе хранения данных: готовая база данных или же прописанная лично структура. В конечном счете, сделан был выбор в сторону второго во избежание дополнительных сложностей с работой базы данных, а также возможных последующих запретов со стороны разработчиков на ее эксплуатацию в связи с территориальным происхождением программы. Также, собственная структура данных предполагает гибкость по отношению к нуждам разработчика и пользователя, и единственный

ее недостаток — это отдельная реализация, поскольку не является готовым инструментом, а требует затраты временных ресурсов на свою реализацию. Было создано несколько классов, а также глобальных листов с типом этих классов для удобства, чтобы обращаться к ним напрямую, а не передавать из метода в метод.

Поскольку большая часть графических объектов создавалась динамически, то их создание было прописано через код на языке программирования C#, тогда как в целом интерфейсные области и кнопки (объекты, которые присутствуют статически) через XAML.

Для хранения информации о дисциплинах и взаимосвязях между ними использовался специальный класс ObservableCollection<T>, по функциональности похожий на список List за тем исключением, что позволяет известить внешние объекты о том, что коллекция была изменена. Хранилище данных являлось глобальной переменной для упрощения взаимодействия с ними методов. В качестве типа информации, хранившейся в коллекции, использовался написанный вручную класс Discipline, включающий в себя свойства названия дисциплины, семестра, комментария, а также наличия списков List<Discipline> как предшествующих ей дисциплин, так и последующих.

При запуске программы появляется статичный интерфейс (рис. 1), где изменения можно будет наблюдать лишь в области 1 и 3. Область 2 остается неизменной.

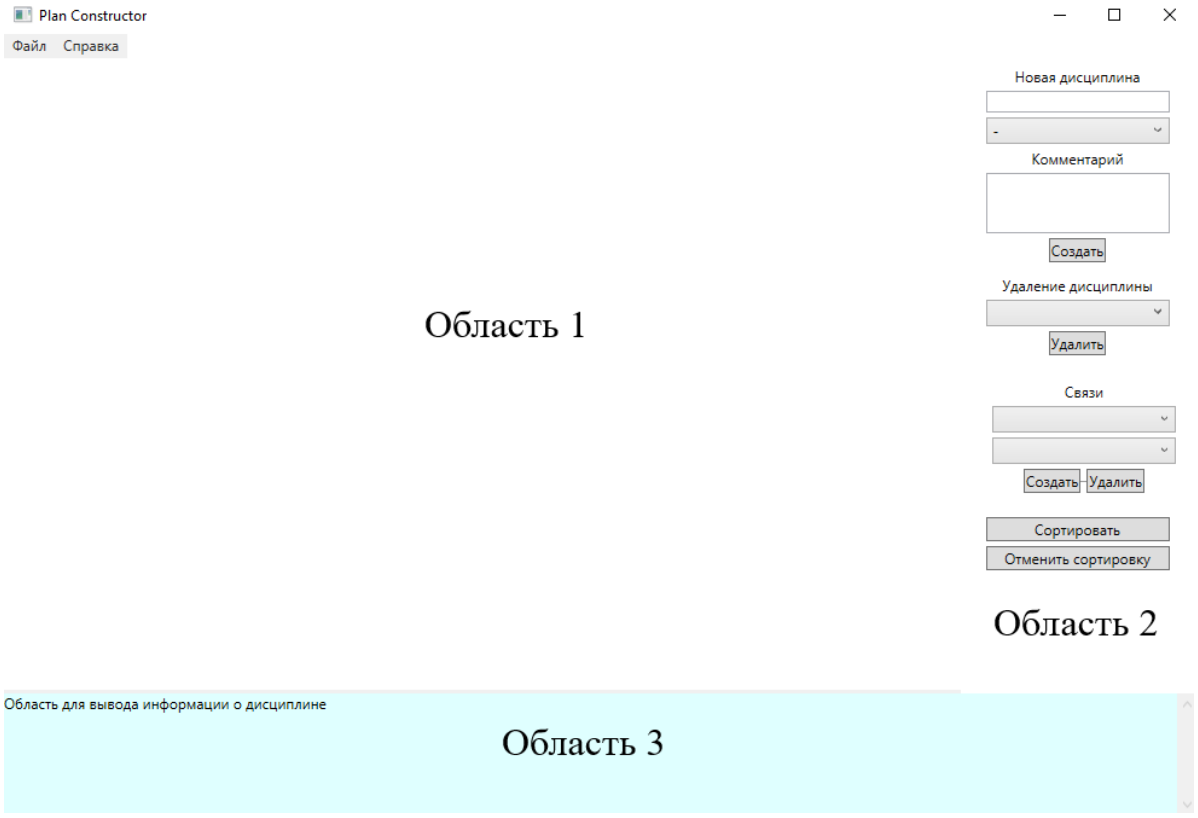

Рис 1. Статичный интерфейс.

Добавление дисциплин происходит через форму в области 2, где вводится ее название, выбирается один из 8 семестров, а также пишется комментарий, после чего пользователь нажимает на кнопку «Создать», либо клавишу Enter. Обязательно лишь поле названия, если программа находится в режиме орграфа. В режиме сортировки она затребует указание семестра (по умолчанию он будет записываться как нулевой). Добавленные дисциплины мгновенно отображаются в области 1 (рис. 2).

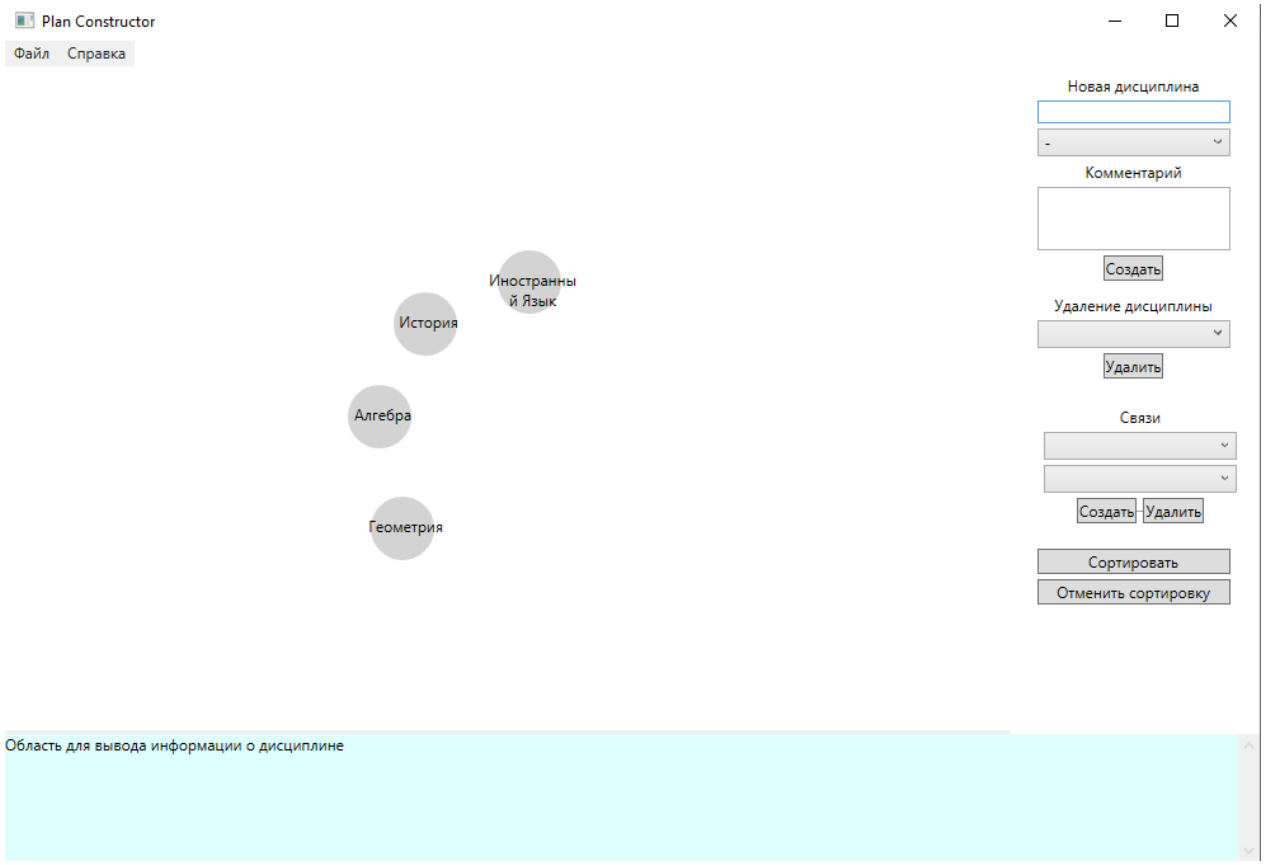

Рис. 2. Добавление дисциплины.

Между дисциплинами могут настраиваться связи. Проектировщику плана может быть точно известно, что одна дисциплина должна следовать за другой или предшествовать третьей. Создание отношения происходит в форме в Области 2 (рис. 3). Из списков зарегистрированных дисциплин выбираются необходимые и создается отношение. Таким же образом связь может и удалиться. При этом, если у предшествующей дисциплины указан семестр больший, чем у последующей, то программа выведет сообщение об ошибке.

Область 1 является интерактивной. При нажатии левой кнопкой мыши на дисциплину, она выделяется (меняет цвет) (рис. 4), а в Области 3 о ней выводится информация. При нажатии правой кнопкой мыши на дисциплину, появляется контекстное меню, предлагающее удаление или же редактирование. Удаление может происходить как через контекстное меню, так и через форму в Области 2.

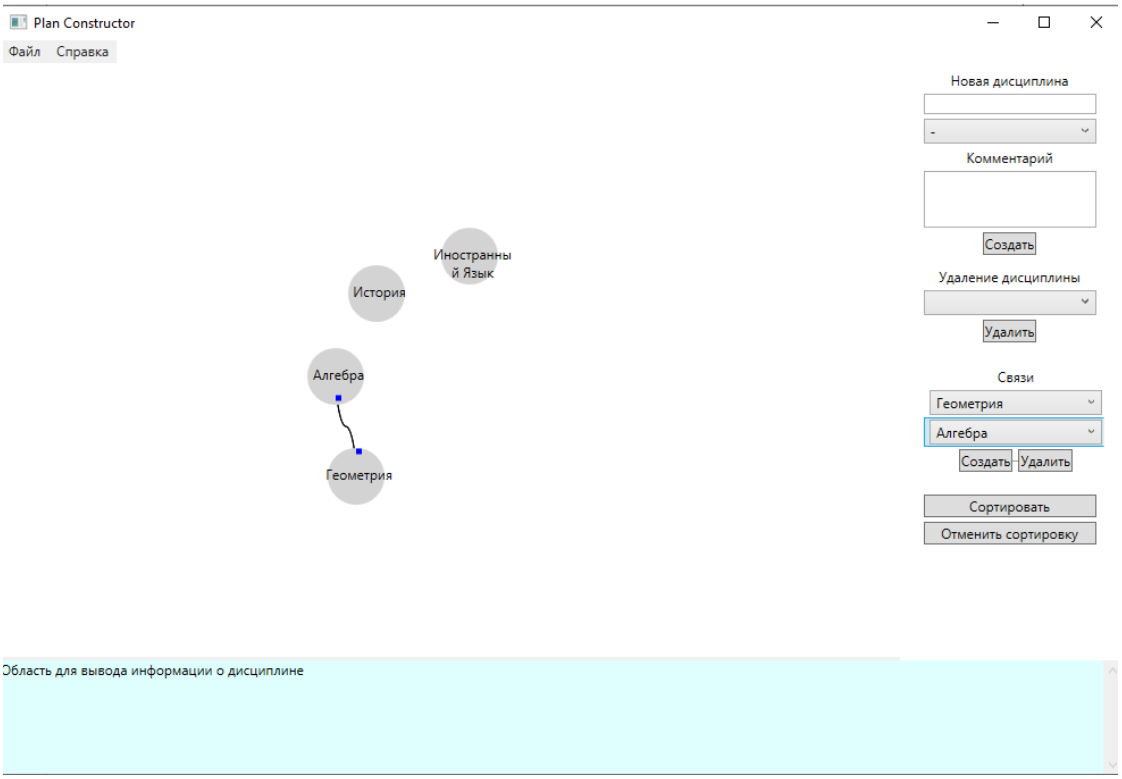

Рис. 3. Добавление связи.

При выборе опции «Редактировать» выводится отдельное окно (рис. 4), обращающееся к глобальным переменным, хранящим запрос. Изменения данных сохраняются и передаются в хранилище, и в основном окне происходит перерисовка графа.

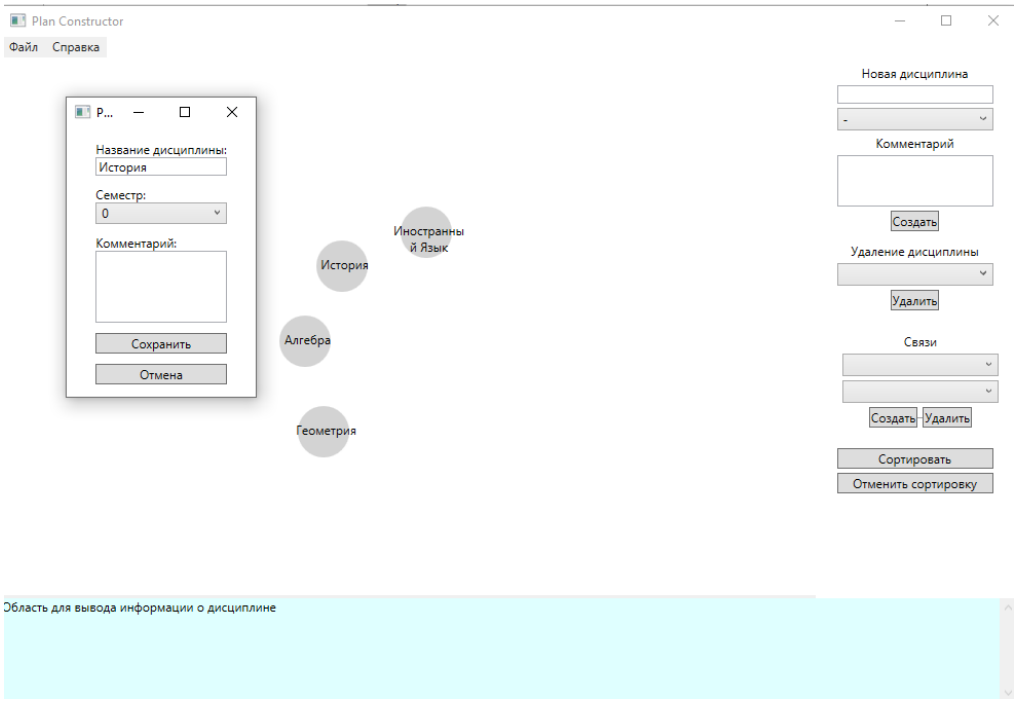

Рис. 4. Выведение окна редактирования.

Программа содержит в себе два режима: ориентированный граф и упорядоченные по семестрам и алфавиту списки (рис. 5, рис. 6). Их различие, помимо визуальной составляющей, заключается в том, что орграф не требует немедленного указания семестра и предполагает редактирование свойств дисциплины, тогда как второй режим подразумевает проверку имеющихся данных на то, чтобы все дисциплины (как уже имеющиеся, так и добавляющиеся) шли с указанием семестра. При том важным аспектом являются взаимосвязи, так как уже на этапе создания отношения между двумя дисциплинами, идет проверка адекватности подобной связи – не поставлена ли предшествующая дисциплина на семестр впереди последующей. Переключение между режимами идет по двум кнопкам – «Сортировать» и «Без сортировки». Их нажатие вызывает методы, которые меняют состояние глобальной переменной, отслеживающей то, какой режим на данный момент активирован в программе.

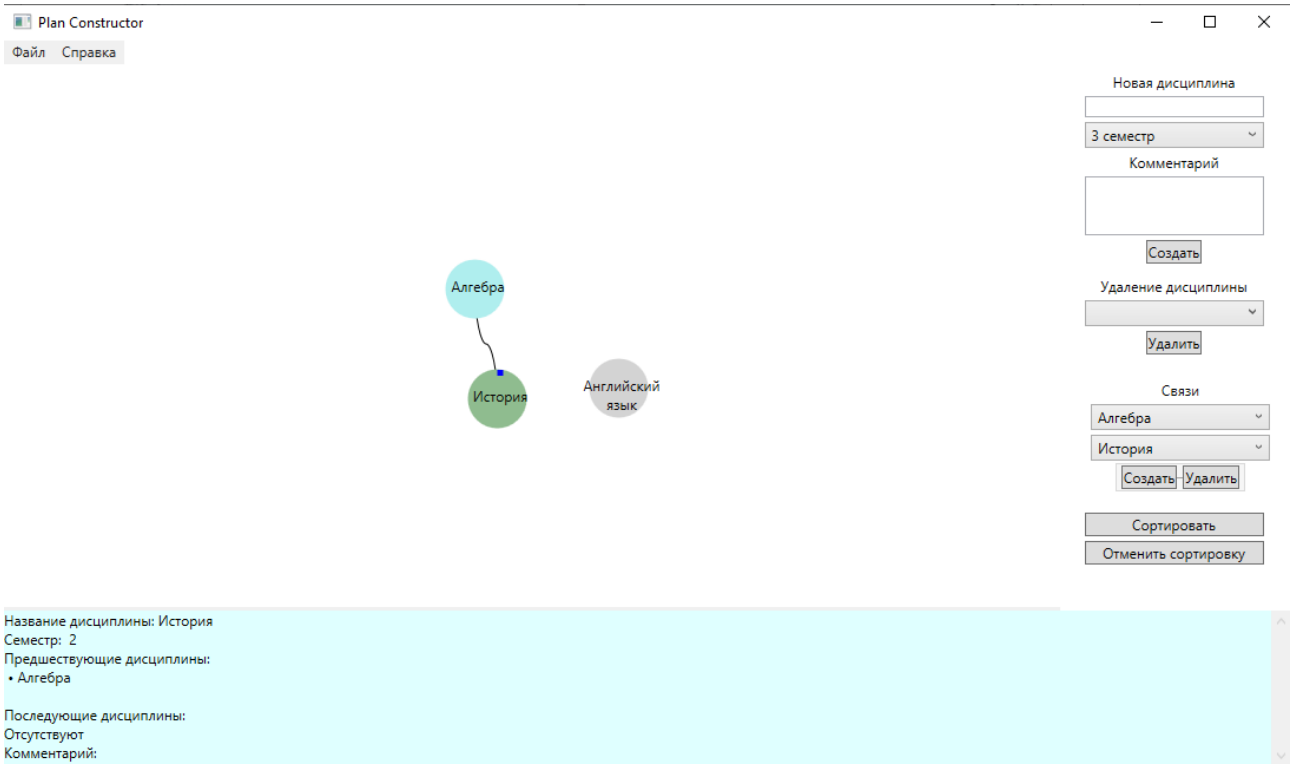

Рис. 5. Режим ориентированного графа.

В зависимости от режима, срабатывают разные методы добавления необходимых компонентов в Canvas (контейнер в WPF). В случае орграфа данные об элементах хранятся в соответствующих глобальных списках, т.к. расположение элементов изначально задается случайным образом, и, если каждый раз при добавлении дисциплины будет идти перерисовка, это негативным образом скажется на эффективности визуализации – если элементы будут постоянно менять свое месторасположение, это может привести к путанице. Поэтому данные о всех графических элементах ориентированного графа хранятся в списках (List<T>), и при

прорисовке новые элементы добавляются в Canvas без необходимости предварительного очищения поля (если только речь не идет о переключении между режимами).

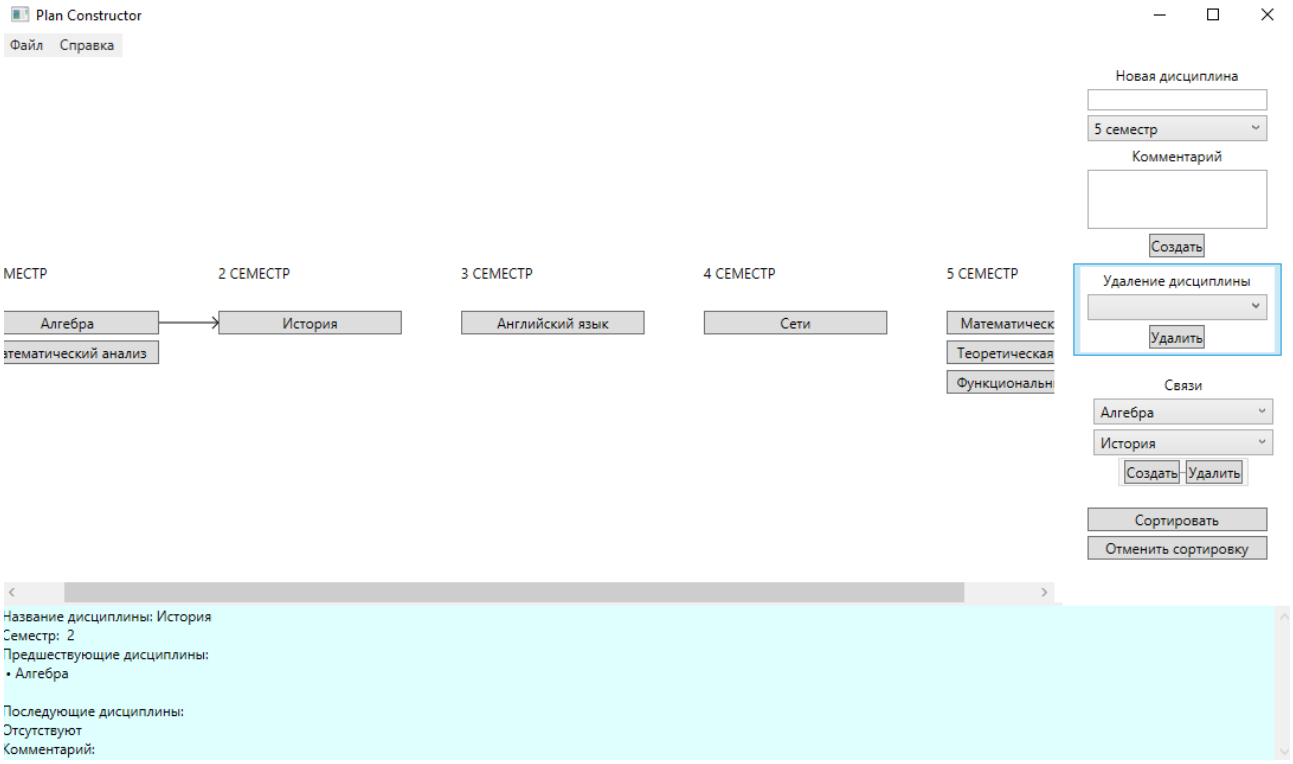

Рис. 6. Режим расписания по семестрам.

В случае режима списков по семестрам при добавлении новых дисциплин без переключения на режим орграфа, удаления или же переключения между двумя состояниями поля каждый раз идет перерисовка. Это осуществляется с той целью, чтобы список всегда был упорядоченным по алфавиту, и между элементами не было ненужных «прогалов», если речь идет об удалении дисциплины. Также расположение визуальных составляющих происходит не случайным образом, а потому нет необходимости в запоминании чьих-либо позиций – достаточно хранить данные о самих дисциплинах (самыми важными свойствами в данном случае будут название и семестр).

Для вызова окна редактирования в режиме орграфа используется новое окно со своими методами. Необходимые данные о дисциплине, которая будет редактироваться, записываются в глобальных переменных, чтобы передавать информацию между окнами (главным окном и окном редактирования). Если же какие-то изменения будут внесены в окне редактирования, то глобальные переменные будут перезаписаны, и пользователь вернется на главное окно, в котором метод перезапишет информацию и внесет необходимые изменения, как в структуры данных, так и в визуальную составляющую.

В конце разработки с соблюдением требований было получено заявленное программное обеспечение для составления учебного плана, позволяющее создать хранилище данных о существующих дисциплинах и их параметрах и прошедшее тестирование.

СПИСОК ЛИТЕРАТУРЫ

1. ММИС Лаборатория: Программы: GosInsp: сайт / Лаборатория ММИС [Электронный ресурс]. – Режим доступа: https://www.mmis.ru/programs/GosInsp (дата обращения: 30.05.2023).
2. Obsidian: сайт / Obsidian, 2023 [Электронный ресурс]. – Режим доступа: https://obsidian.md/ (дата обращения: 30.05.2023).
3. Internet Archive WayBackMachine: Основы Программной Инженерии (по SWEBOK): сайт / "Основы программной инженерии" Copyright, Сергей Орлик, 2004-2010 [Электронный ресурс]. – Режим доступа: https://web.archive.org/web/20100604013037/ http://swebok.sorlik.ru/1_software_requirements.html (дата обращения: 27.10.2023).