СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Потребность в записи информации на различные носители, её быстром изменении и надёжном хранении была и остаётся важной в повседневной жизни для любого человека. Эта потребность играла немаловажную роль в обществе с самого его зарождения, и теперь, в эру информационных технологий, она всё усиливается, приобретая особую значимость. С появлением более совершенных носителей информации, способов её сохранения и изменения, открываются обширные возможности для реализации технологий, позволяющих решить задачу эффективной обработки информации. Наиболее распространённой и востребованной её формой является текст, именно поэтому различные приложения для работы с текстом являются наиболее популярными. Среди этих приложений особую нишу занимают программы для ведения заметок. Существует множество программ из этого класса, которые пользуются немалым успехом у пользователей. Крайне популярными являются, например, такие программы, как Microsoft Office OneNote, Evernote или Google Keep. Эти приложения позволяют быстро и эффективно обрабатывать небольшие объёмы текста, что делает их крайне удобными и эффективными в повседневном использовании.

Целью данной курсовой работы является разработка программы, выполняющей функции записной книжки, в парадигме Объектно-ориентированного программирования. В ходе выполнения работы следует решить следующие задачи:

1. Декомпозиция задачи.
2. Построение диаграмм, описывающих приложение.
3. Создание приложения, обладающего необходимыми функциями и понятным пользовательским интерфейсом, с применением объектно-ориентированного подхода.
4. Тестирование и отладка приложения.

В результате выполнения поставленных целей и задач должно быть создано приложение «Записная книжка».

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Назначение и область применения

Предназначением разрабатываемого приложения является предоставление пользователям возможности быстрого и эффективного просмотра, создания, редактирования, удаления и сохранения коротких заметок и записок. Программа должна обладать простым и доступным интерфейсом, который будет понятен любому пользователю, даже не имеющему опыта в области эксплуатации какого-либо программного обеспечения для работы с текстом. Программа должна использоваться на персональном компьютере.

Область применения разрабатываемого проекта достаточно широка. Он может применяться как для офисной работы, так и для работы на каком-либо предприятии. Однако основной областью применения, несомненно, является использование данной программы пользователем в домашних, бытовых условиях, для быстрой записи какой – либо информации с целью не потерять данную информацию в дальнейшем.

1.2. Описание программы

Главной особенностью программы такого рода должен быть простой и понятный интерфейс. В разрабатываемом приложении его основой должно стать поле, в котором будет находиться отображаемый текст заметки. Программа должна позволять редактировать данный текст и сохранять его. Сохранение должно производиться после нажатия на клавишу сохранения.

Помимо этого, интерфейс должен включать список существующих заметок, при нажатии на элемент которого соответствующая заметка должна выводиться на экран, а её название должно отображаться в соответствующем заголовке. Также должна присутствовать возможность добавления и удаления выбранной заметки с помощью нажатия на соответствующие клавиши в окне программы. В дополнение к

указанным выше функциям, должна присутствовать возможность изменить название какой-либо заметки.

Основой работы программы должны стать файлы, организованные по специальному принципу. Каждая заметка будет храниться в специальном, соответствующем ей текстовом файле. Для общего списка заметок будет создан особый текстовый файл, также ассоциированный с данным списком. Этот файл будет иметь специальное имя и расширение. При запуске программы информация из файла, соответствующего списку заметок, будет считываться, и список будет отображаться. При переименовании какой-либо заметки, файл со списком будет перезаписываться, и, таким, образом, изменения будут сохраняться. Помимо этого, при переименовании заметки, соответствующий ей файл также будет переименован.

При нажатии на какой-либо элемент из списка заметок, файл, соответствующий данному элементу, будет считываться, и его содержимое будет отображаться в текстовом поле, а заголовок будет меняться на название выбранной заметки, совпадающее с названием считанного файла. При изменении текста в текстовом поле, файл, соответствующий текущей заметке, будет перезаписываться, и, таким образом, изменения в заметке будут сохраняться. В случае удаления выделенного элемента из списка, файл, ассоциированный с ним, также будет удаляться с диска.

1.3. Выбор инструментальных средств

Реализуемый проект должен быть написан в парадигме объектно-ориентированного программирования. Объектно-ориентированное программирование – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования. Объектно-ориентированный подход к программированию позволяет разложить задачу на составные части таким образом, что каждая составная часть будет представлять собой самостоятельный объект, который содержит собственные инструкции и данные.

Концепция объектно – ориентированного программирования позволяет программисту, концентрируясь на несложных составных частях одной сложной задачи, таким образом повышать производительность и решать достаточно трудоёмкие задачи, решение которых было бы затруднено при использовании парадигмы структурного программирования. Именно объектно –ориентированный подход к программированию является доминирующим в настоящий момент. Его реализацию поддерживают такие языки программирования, как C++, C#, Java и т.д.

В качестве языка программирования для решения данной задачи был выбран язык C++. Данный язык является языком высокого уровня, который сохраняет лидерские позиции среди других языков на протяжении уже достаточно долгого промежутка времени. C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений. Также данный язык постоянно обновляется и совершенствуется, что позволяет ему сохранять актуальность.

В качестве фреймворка для создания приложения был выбран Qt — кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++. Данный фреймворк обладает широким набором возможностей для создания программного обеспечения. Он позволяет быстро и эффективно проектировать, разрабатывать, выпускать и сопровождать программные продукты. Более того, Qt содержит такой инструмент, как Qt Designer, позволяющий создавать интерфейс программы, что значительно облегчает процесс его создания.

2.1 Реализация программы

Перед непосредственным созданием приложения в среде разработки была проведена декомпозиция задачи, с помощью которой были выявлены основные этапы разработки приложения.

Были построены диаграммы, позволяющие более чётко и ясно понять структуру программы и её внутреннее устройство.

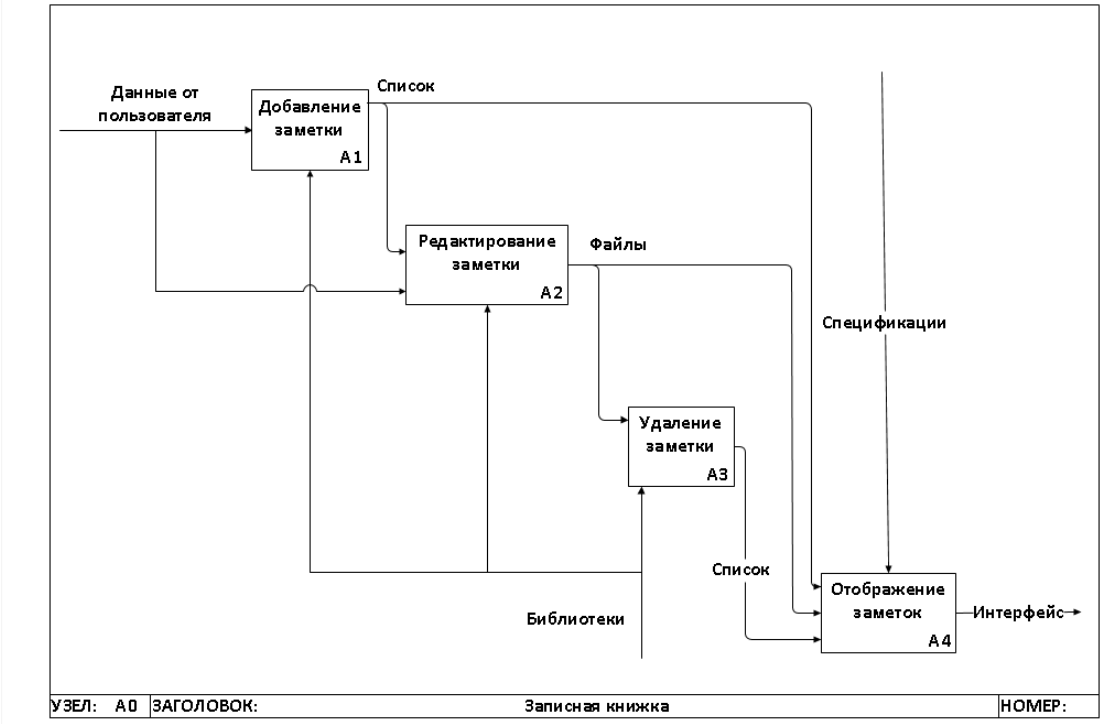


Рисунок 2.1 Диаграмма Idef0

После построения диаграмм было начато непосредственное создание программы в Qt Creator. Был создан класс MainWindow соответствующие ему файлы с расширениями cpp, h и ui. В первую очередь был создан интерфейс программы в Qt Designer путём редактирования файла с расширением ui. Основное окно было переименовано, и с него было убрано всё лишнее, после чего на него были добавлены:

1. Область для редактирования текста (Text Edit) с названием textEdit (для отображения и редактирования текущей заметки)
2. Список (List Widget) с названием listWidget (для показа и изменения списка всех заметок)
3. Кнопка (Tool Button) с названием toolButton (для добавления заметки в список)
4. Кнопка (Tool Button) с названием toolButton_2 (для удаления заметки)
5. Кнопка (Tool Button) с названием toolButton_3 (для сохранения заметки)
6. Метка (Label) с названием label (для обозначения над списком заметок)
7. Метка (Label) с названием label_2 (для показа названия текущей заметки)

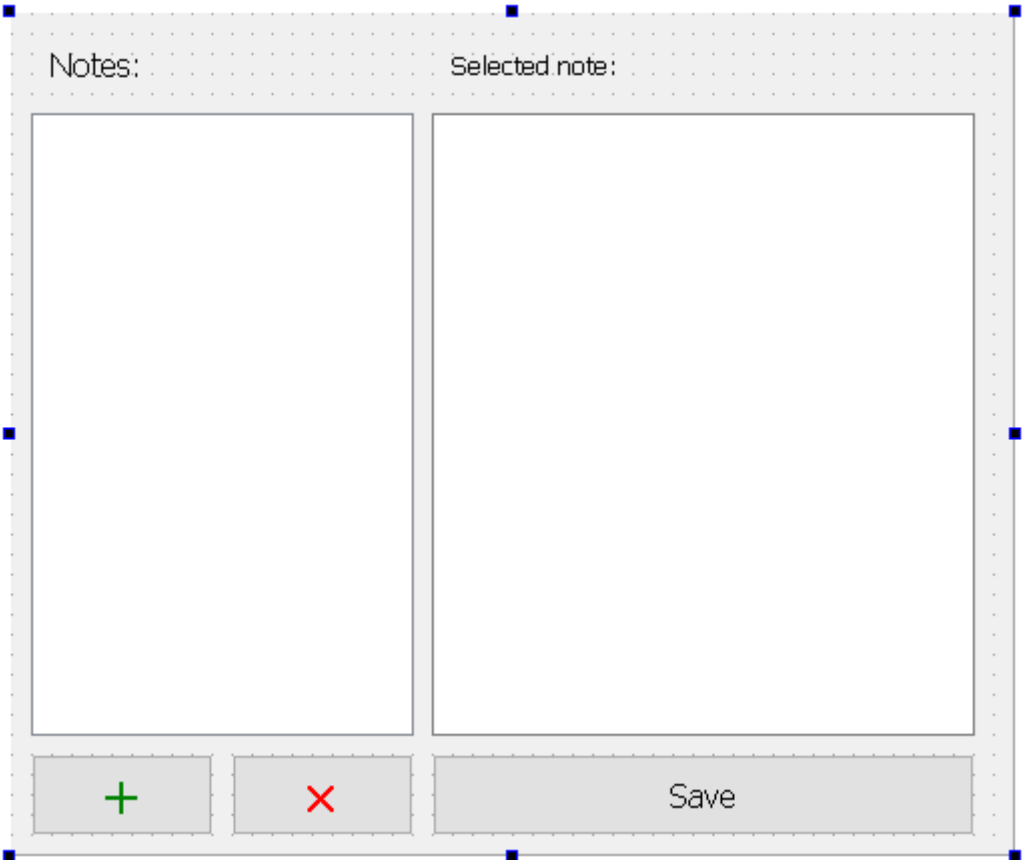


Рисунок 2.2 Создание интерфейса программы

После добавления всех кнопок и других виджетов на форму и необходимого изменения их внешних свойств, было начато непосредственное программирование действий для этих элементов.

В файле mainwindow.h были подключены следующие библиотеки:

- QFile – этот класс предоставляет интерфейс для чтения и записи в файлы. Данный класс нужен для того, чтобы сохранять, изменять или удалять файлы с заметками. Также он содержит некоторые полезные функции, используемые в работе программы, такие как, например, bool QFile::exists() const, bool QFile::remove(const QString & fileName) или bool QFile::rename(const QString & oldName, const QString & newName).
- QTextStream – класс, который обеспечивает удобный интерфейс для чтения и записи текста. Он необходим, чтобы записывать заметки в файл и считывать их оттуда.
- QListWidgetItem – класс, который предоставляет элемент для использования с классом представления элементов QListWidget. Он необходим для того, чтобы добавлять, удалять или изменять элементы списка, соответствующие заметкам.

В конструктор было добавлено считывание списка заметок из соответствующего ему файла и добавление заметок в список для отображения.

Для сигнала clicked() кнопки toolButton был добавлен слот void MainWindow::on_toolButton_clicked(), который создаёт новый элемент и добавляет его в список, перезаписывая файл со списком.

Для сигнала clicked() кнопки toolButton_2 был создан слот void MainWindow::on_toolButton_2_clicked(), задачей которого является удаление текущего элемента из списка, удаление файла, соответствующего элементу списка, и перезапись файла списком.

Для сигнала clicked() кнопки toolButton_3 был создан слот void MainWindow::on_toolButton_3_clicked(), цель которого состоит в создании файла с заметкой и сохранении в нём введённого текста, либо, если текущая заметка уже

была создана, в перезаписи файла этой заметки с целью сохранить изменённый текст.

Для сигнала itemClicked(QListWidgetItem*) списка listWidget был создан слот void MainWindow::on_listWidget_itemClicked(QListWidgetItem *item), который должен считывать выделенную в списке заметку из файла и отображать её в текстовом поле.

Для сигнала itemChanged(QListWidgetItem*) списка listWidget был создан слот void MainWindow::on_listWidget_itemChanged(QListWidgetItem *item), который отрабатывает при изменении элемента списка. Этот слот необходим, чтобы сохранять изменения в элементах списка заметок, в первую очередь, в случае их переименования. Он переименовывает файл, соответствующий переименованной заметке, и сохраняет список с переименованной заметкой в файл, соответствующий ему. Этот слот также сохраняет в файл со списком заданное пользователем имя новой заметки.

Для сигнала itemDoubleClicked(QListWidgetItem*) списка listWidget был создан слот void MainWindow::on_listWidget_itemDoubleClicked(QListWidgetItem *item), который выделяет и отображает заметку, на которую в списке кликнули дважды, и переводит название этой заметки в списке в режим редактирования. В дальнейшем, после задания пользователем нового имени для заметки, вызывается сигнал itemChanged, который сохраняет внесённые изменения.

В результате выполнения вышеуказанных действий было создано работоспособное приложение. После этого был начат этап тестирования.

2.2 Тестирование

После успешного написания кода программы, был начат процесс отладки разрабатываемого приложения. В ходе тестирования в нём создавались, редактировались, переименовывались, сохранялись и удалялись заметки. Также отслеживалось, корректно ли создаются, удаляются, сохраняются и изменяются файлы на диске, которые данное приложение должно обрабатывать.

В случае возникновения неполадок, ошибок и некорректной работы приложение дорабатывалось, а ошибки в коде исправлялись. В результате было получено полностью работоспособное и стабильное приложение, успешно выполняющее все возложенные на него задачи.

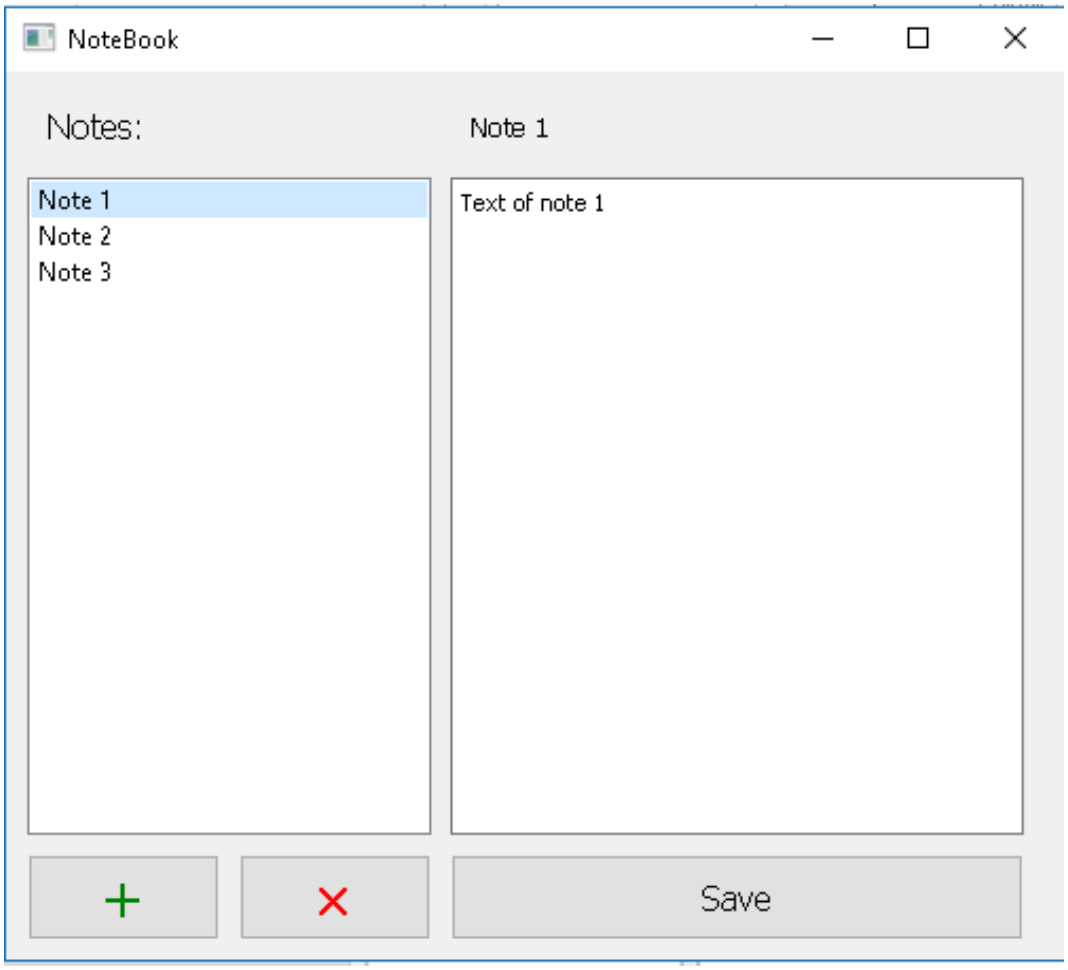


Рисунок 2.3 Готовая программа

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было создано приложение «Записная книжка», позволяющее эффективно создавать, просматривать, редактировать, удалять и сохранять заметки. Приложение обладает простым и интуитивно понятным интерфейсом и доступно для использования любому пользователю.

В ходе выполнения курсовой работы были решены следующие задачи:

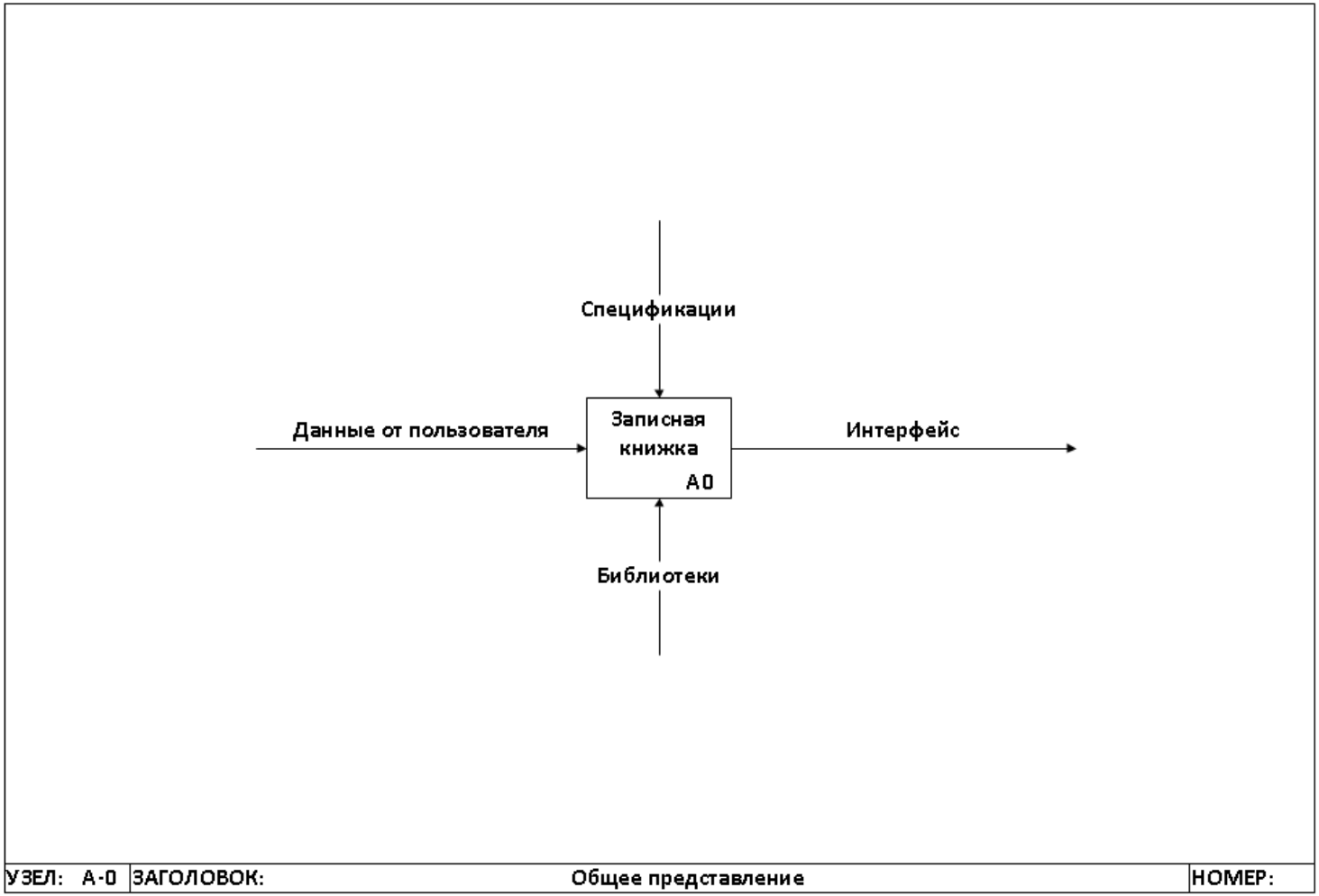
- Была проведена декомпозиция задачи, были выделены основные этапы разработки приложения.
- Были созданы диаграммы, поясняющиея внутреннее устройство приложения.
- С применением объектно-ориентированного подхода был написан код программы на языке C++, удовлетворяющей всем поставленным требованиям и обладающей всеми необходимыми функциями.
- Было проведено тестирование и отладка программы, в ходе которой были выявлены и устранены ошибки в приложении. Удалось добиться стабильной и корректной работы приложения.

Итак, все задачи, возникшие при выполнении курсовой работы, были решены, а поставленные цели достигнуты.

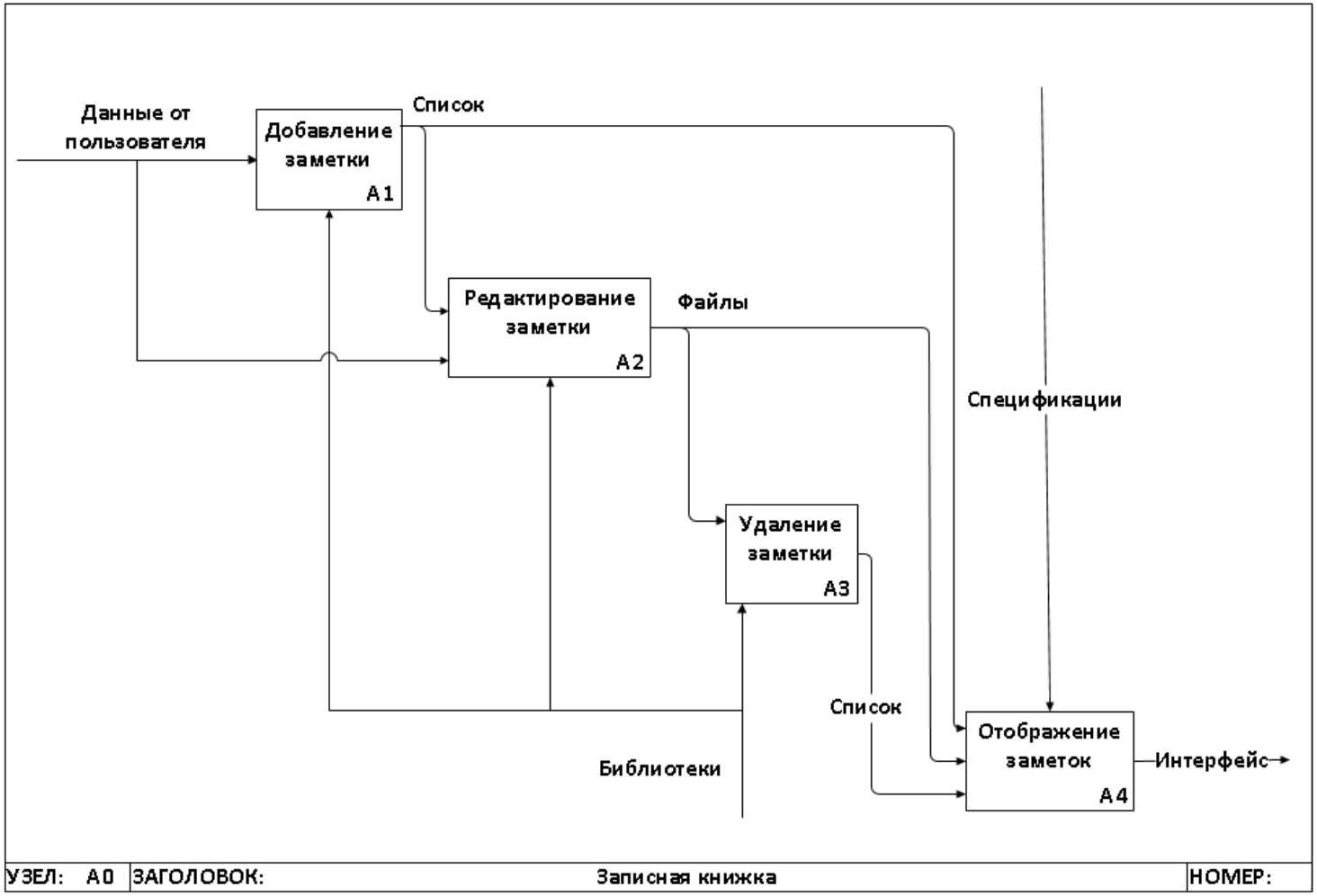
СПИСОК ИСТОЧНИКОВ

1. Шилдт, Г. C++: Базовый курс, 3-е издание. : Пер. с англ. – М.: Издательский дом «Вильямс», 2011. – 624 с.
2. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. – 2-е изд., перераб. и доп. – М.: Финансы и статистика, 2006. – 544 с.
3. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. — 3-е издание. — «Вильямс», 2010.- 718 с.
4. Qt Documentation: [Электронный документ]. – (http://doc.qt.io/). – Проверено 18.12.2017.
5. Макс Шлее. Qt 4.8 Профессиональное программирование на C++. — СПб.: БХВ-Петербург, 2012. — 912 с.

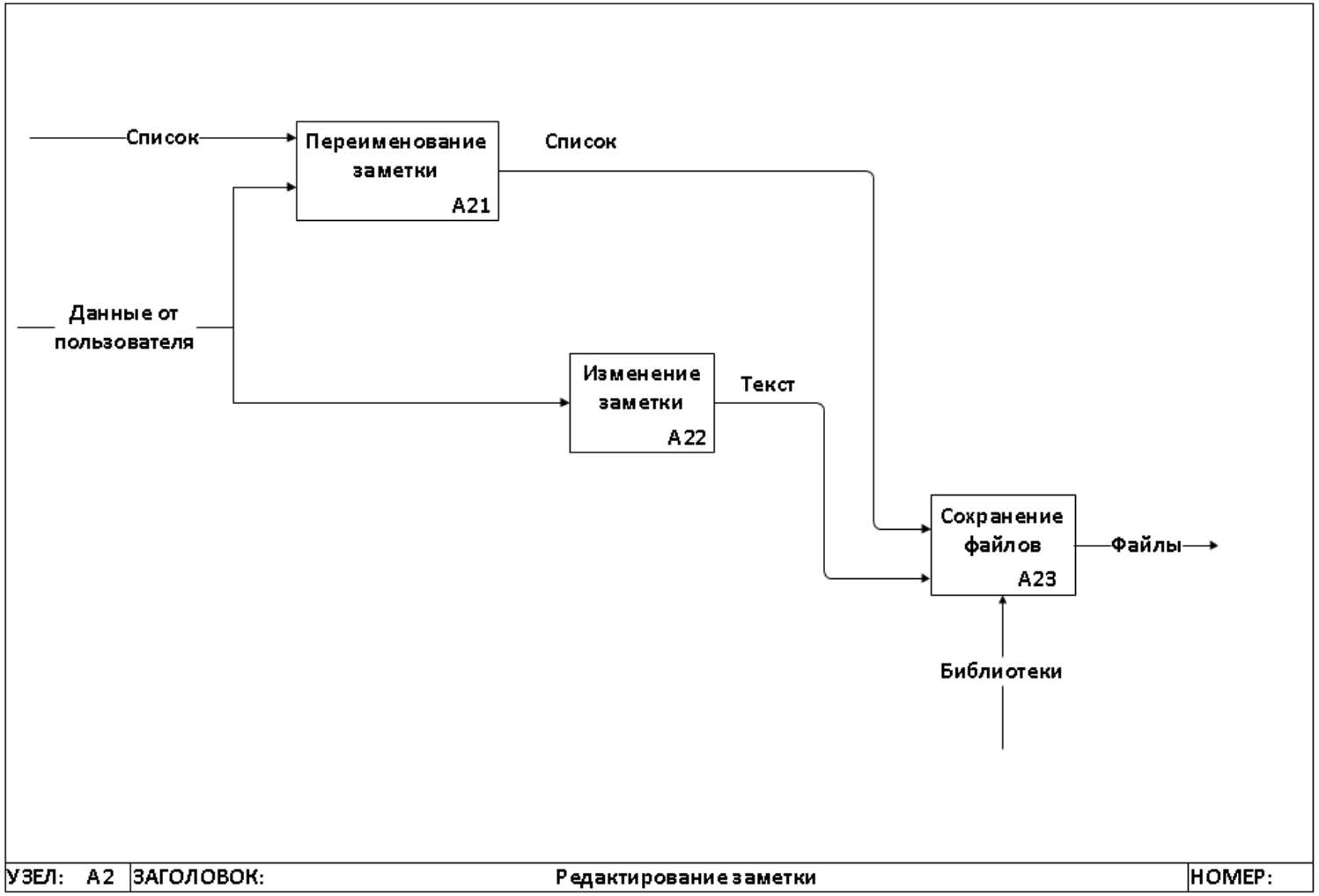
ПРИЛОЖЕНИЕ 1



ПРИЛОЖЕНИЕ 2



ПРИЛОЖЕНИЕ 3



ПРИЛОЖЕНИЕ 4