Содержание:

ВВЕДЕНИЕ

Все мы так или иначе взаимодействуем с различными интерфейсами: кто – то пытается заказать себе кофе через интерактивный экран в Макдональдсе, кто – то ищет книги в библиотеке с помощью современных технологий, кто – то пишет статью на своем ноутбуке или проверяет почту с планшета и телефона.

Пользовательский интерфейс, или интерфейс, - это способ взаимодействия с вашим устройством. Сначала вам приходит идея какого – то продукта, который должен так или иначе помочь облегчить жизнь человека. После этого начинается разработка сценария, где вы продумываете все варианты действий, которые может совершить пользователь с вашим продуктом. Пользователь становится героем сказки, в которой есть множество сюжетных ответвлений, начинающихся со слов: «А что, если? А что, если я нажму эту кнопку? А что, если я поверну экран? А что если?».

Каждый из этих вариантов должен быть продуман, чтобы перейти к этапу проектирования. Но за кучей «что если» разработчики иногда совершенно забывают о том, что их конечные пользователи, нагружают продукт и делают его настолько сложным, что решений этой задачи, для которой он должен пригодится, даже рядом не стоит. Из –за колоссальной конкуренции на рынке ПО, как правило, имеется немало аналогов практически любого программного продукта и интерфейсами на любой вкус. Пользователей можно понять, ведь время является одним из наиболее ценных ресурсов, а это время, которое пользователь тратит на ознакомление и изучение программы, могло бы быть потрачено для решения поставленной задачи. Поэтому одним из основных правил проектирования интерфейса пользователя является минимизация времени выполнения задачи пользователя. Программа должна помогать и упрощать работу пользователя, а ни в коем случае не затруднять ее.

Актуальность данной курсовой работы определяется тем, что в настоящее время в разработке находится большое количество приложений, связанных с распределенной обработкой данных, и постоянно тратить время на решение одних и тех же задач, направленных на эту обработку, является бессмысленным. Исходя

их этого, становиться актуальной задача, связанная с повышением эффективности разработки, ведь решение этой задачи позволит программистам сократить время разработки будущих приложений с обработкой данных на отдельном сервере.

В основу информационной базы данного исследования легли работы российских и иностранных исследователей в сфере проектирования и построения пользовательских интерфейсов таких как: Горнец Н.Н.,[1]Керниган Б.У., [2]Киселев Г.М.,[3] и др.

Объектом исследования является формирование информационно – предметной среды пользовательского интерфейса.

Предметом исследования является методика проектирования Фреймворка.

Целью данной курсовой работы является – проектирование Фреймворка для построения распределенной информационной системы в процессе разработки приложения Bookstore.

Для достижения поставленной цели в ходе исследования необходимо решить ряд задач:

- Изучить теоретические аспекты пользовательских интерфейсов;
- Рассмотреть этапы развития и типы пользовательских интерфейсов;
- Изучить эволюцию графических интерфейсов операционных систем;
- Раскрыть понятие Фреймворка;
- Проанализировать инструментальные средства проектирования, выбрать оптимальные из них;
- Спроектировать Фреймворк;
- Провести апробацию готового Фреймворка.

Курсовая работа состоит из введения, двух взаимосвязанных глав, заключения, списка использованных источников.

1.ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ОСОБЕННОСТЕЙ ПРОГРАММ ИНТЕРФЕЙСОВ И ИХ ЭВОЛЮЦИЯ

Этапы развития и типы пользовательских интерфейсов

Интерфейс пользователя (**user interface** или сокращенно далее **UI**) – это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. **UI** должны быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне. Интерфейсы программного обеспечения также называют графическими пользовательскими интерфейсами (**graphical user interface или GUI**).[4]

Пользовательский интерфейс – это средства взаимодействия между человеком и компьютером. Говоря простыми словами, интерфейс – внешняя часть программы или устройства, с которыми работает пользователь. Слово интерфейс – калька с английского interface, то есть «граница, связующее звено».[5] Чаще всего под словом интерфейс подразумевают именно пользовательский интерфейс. Например, говорят: «У этого Интернет – магазина неудобный, запутанный интерфейс». Это означает, что с сайтом магазина неудобно взаимодействовать. Скажем, сложно найти нужные товары, непонятно, как оформить заказ, сайт не сохраняет ранее введенные данные и т.п.

Веб – интерфейс (web – interface) – это страница в Интернете, позволяющая пользователю взаимодействовать с каким – то сервисом или устройством прямо через браузер. К примеру, с помощью веб – интерфейса можно воспользоваться онлайн – банком: зайти на страницу банка, ввести логин и пароль, а затем переводить деньги между счетами, оплачивать услуги и т.п.

Пользовательский интерфейс (далее ПИ) - это все, с чем взаимодействует пользователь при работе с устройством. В широком смысле, термин можно отнести ко всем устройствам, с которыми работает человек — от персональных компьютеров до сложных промышленных систем. Однако в наше время, говоря «пользовательский интерфейс», как правило, имеют в виду интерфейсы устройств, для работы с которыми не требуется специальная профессиональная подготовка (компьютеры, бытовые приборы, мобильные телефоны и планшеты, банкоматы и терминалы и т.п.).[6] Типичный ПИ имеет устройства ввода и вывода. Получив от пользователя команду, интерфейс «отвечает» ему, выводя разного рода информацию. В качестве примера можно привести цифровой термометр: нажав на кнопку (устройство ввода), пользователь запускает процесс измерения

температуры, и видит результат на дисплее (устройство вывода).

В подавляющем большинстве случаев под термином «пользовательский интерфейс» имеется в виду графический интерфейс пользователя (graphical user interface, GUI) — разновидность пользовательского интерфейса, в котором человек взаимодействует с системой при помощи графических компонентов (окна, пиктограммы, полосы прокрутки и т.п.), отображаемых на экране.[7]
В различных компьютерных играх применяется натуральный пользовательский интерфейс (NUI или natural user interface). Его система анализирует движения человека, и преобразует их в движения в игре. На данный момент в стадии разработки находится перцептивный пользовательский интерфейс (PUI), а также интерфейс мозг-компьютер (BCI или brain-computer interface). Последняя разработка направлена на то, чтобы обеспечить людям возможность управлять компьютерами силой мысли.

Помимо пользовательского интерфейса существуют программный интерфейс (взаимодействие программ между собой) и аппаратный интерфейс (способы взаимодействия физических устройств, «железа»).[8]

Когда говорят об аппаратном интерфейсе, обычно имеют ввиду разъемы, через которые устройства можно подключить друг к другу. Например, «подключение через интерфейс USB» - это значит соединение устройств через универсальную последовательную шину, предназначенную для подключения переферийной техники. Через USB, например, можно подключить к компьютеру клавиатуру, мышку, фотоаппарат или смартфон.

Программный интерфейс - это способ взаимодействия программ между собой. Например, API (application programming interface, программный интерфейс приложения) - это набор команд, который позволяет программам автоматически обмениваться данными без участия людей. Одна программа по API отправляет запрос, другая отвечает ей. К примеру, на новостном сайте показываются курсы валют, которые меняются в реальном времени. Это не значит, что редактор сайта каждый раз вручную меняет числа на странице. Новостной сайт сам отправляет по API запрос на сервер с данными валютной биржи и получает оттуда необходимые цифры.[9]

К основным типам пользовательских интерфейсов относятся:

Текстовый интерфейс - это способ общения человека с компьютером с помощью печати команд. Например, в операционной системе MS-DOS интерфейс был

текстовым - пользователь набирал на клавиатуре нужные команды, а машина их выполняла. Этот тип интерфейса пользователя предназначен для работы с символами. Исполнение происходит в режиме аппаратного текста, однако часто используется и дисплей. В данном случае на каждый источник у программиста имеется 256 символов. Навигация производится клавиатурой, а не мышью. В качестве примера можно привести **Norton Commander** или**Turbo Pascal**. Norton Commander - файловый менеджер для MS-DOS. В нем можно не только набирать команды на клавиатуре, но работать с файлами с помощью сочетаний клавиш.[10] Этот интерфейс также используется в загрузчиках **ОС** и **BIOS-программах**. Данный тип интерфейса также используется для установки операционных систем. Рисунок 1.

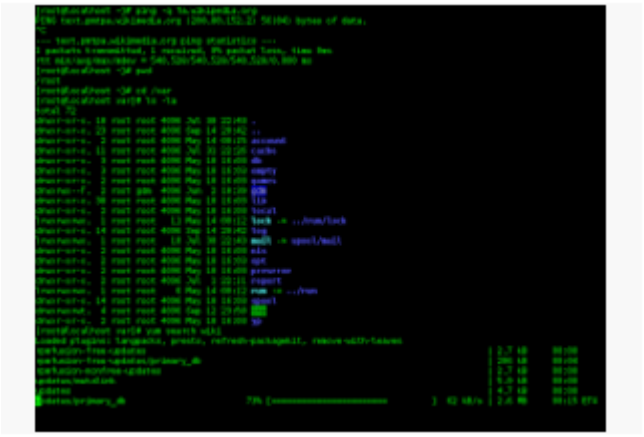


Рисунок 1. Интерфейс командной строки.

Проблема текстового интерфейса в том, что пользователь должен знать необходимые команды и каждый раз вручную набирать их без ошибок. Частично от этой трудности избавили оболочки для MS-DOS - например, Norton Commander.

Графический пользовательский интерфейс. Графический пользовательский интерфейс является наиболее популярным **UI**. Он представляет собой окно, в котором содержатся различные элементы управления. Взаимодействие пользователя с программой при помощи мыши и при помощи клавиатуры. Также есть возможность использовать кнопки и разделы меню, расположенные внутри самого приложения. Это окно представляет собой нечто вроде шлюза между пользователем и программным обеспечением.[11] В графическом интерфейсе пользователя распространены типичные элементы управления. Они позволяют стандартизировать процесс взаимодействия с различными программами в разных операционных системах. Рисунок 2.

Рисунок 2. Низкоуровневые макеты интерфейса калькулятора КАСКО

При разработке первого графического пользовательского интерфейса за основу были взяты элементы реального мира: мусорная корзина, папка, изображение дискеты в качестве кнопки сохранения. Сегодня многие иконки считаются устаревшими, но все равно используются. Даже при использовании современных изображений и иконок дизайнеры стараются хотя бы минимально отразить их предназначение. Это позволяет облегчить интуитивное взаимодействие с интерфейсом. Цель **GUI** заключается в том, что люди могли легко определить предназначение каждой кнопки. Благодаря этому нам не приходится запоминать все команды, как это было в случае с командной строкой.[12]

При разработке **GUI** применяются определенные своды правил, которые помогают сделать программы удобнее в использовании. В качестве примера можно привести 8 золотых правил от **Бена Шнайдермана**. Ниже приведем несколько сносок из этих правил:

- **Согласованность:** взаимодействие должно происходить всегда похожим образом. То есть, следует избегать использования панелей управления с опциями типа “скопировать выделенную область”, “удалить выделенную область”, “добавить выделенную область”. Данный пример показывает

отсутствие согласованности в **GUI**, чего следует избегать;

- **Информативная обратная связь:** все действия, производимые пользователем, должны быть подкреплены обратной связью. Например, если двойной клик открывает программу, то человеку приходится подождать пару секунд, прежде чем он сможет пользоваться этой программой. Чтобы пользователь знал, что его действия принесли результат, нужно проинформировать его об этом. Это можно реализовать сменой курсора. Один из старейших и привычных примеров – это курсор с песочными часами в **Windows**;
- **Не перегружайте память пользователей:** пользователи не в силах запомнить все и сразу. В длинных сегментах взаимодействия, где пользователь вынужден переходить по нескольким окнам, информация всегда должна отображаться в одной и той же области. Менее востребованная информация, которая отображалась в самом начале, должна быть скрыта.[13]

Голосовой интерфейс - это управление с помощью речевых команд. Человеческий голос сегодня умеют понимать даже мобильные телефоны. В этом типе интерфейсов пользователя взаимодействие между пользователем и компьютером происходит с помощью голоса.[14] Например, пользователь может вербально выбрать человека из ранее составленного списка контактов и совершить звонок. Программы для интерпретации речи в текст и для распознавания речи также используют аудио-интерфейсы. Преимущество данной формы взаимодействия заключается в том, что пользователям не нужно ничего, кроме голоса. Текстовый ввод на устройствах обычно усложняется маленькой клавиатурой (на смартфонах с маленьким экраном), и многим зачастую проще продиктовать текст сообщения.

Среди примеров можно отметить голосового помощника **Apple**, **Siri**, **S-Voice** у **Samsung** или голосовой поиск **Google**. Одна из главных задач при проектировании этого интерфейса пользователя (аудио-интерфейсов) заключается в том, чтобы предоставить аудитории комфортные условия для взаимодействия. То есть, при использовании голосовых синтезаторов в техподдержке, важно не обременять клиентов длинными сообщениями.

Тактильный интерфейс позволяет пользователю испытывать осязательные ощущения (нажим, вибрацию и т.п.) и взаимодействовать с компьютером с их помощью.[15] В них взаимодействие происходит за счет применения мячей или других физических объектов. Сегодня данный тип интерфейсов редко используется в повседневной жизни. Если рабочий компьютер постоянно стоит на одном столе, применение тактильных интерфейсов приобретает новый смысл,

однако чаще всего они просто неприменимы в повседневной жизни. Музеи и выставки – отличный пример сферы применения **TUI**. Физическое взаимодействие запоминается лучше любого другого. Кроме этого тактильные интерфейсы дают простор реализации объектов: форма, фактура, цвет. От песочницы с деревянными кубиками до увеличительного стекла для изображений – возможно практически все. Рисунок 3.

Рисунок 3. Тактильный пользовательский интерфейс для незрячих.

Материальный интерфейс - это способ взаимодействия с компьютером с помощью осязаемых конструкций. Например, компьютерная мышка или джойстик - это материальный интерфейс. Двигая мышку по столу, мы одновременно перемещаем стрелку курсора по экрану.

Жестовый интерфейс. Интерфейс призван предоставить пользователю естественный и интуитивный опыт взаимодействия с устройством или программным обеспечением. В то же время, сам интерфейс будет видимым, например, на сенсорном экране.[16] При помощи **NUI** команды пользователя вносятся с помощью жестов и прикосновений. Данный тип интерфейса пользователя также можно комбинировать с **VUI**. Благодаря прямому отклику устройства взаимодействие происходит естественней, нежели при вводе мышью

или клавиатуры. Кроме сенсорных устройств **NUI** также можно использовать в игровых приставках. К примеру, **Nintendo Wii** позволяет воспроизводить действия на экране за счет перемещения контроллера рукой. Среди других примеров – дополнение **Kinect** к **Xbox**, которое позволяет управлять игровым персонажем на экране движениями собственного тела. Что делает взаимодействие более натуральным.

Нейронный интерфейс. Позволяет передавать команды с помощью вживленных в мозг электродов. Двунаправленные нейронные интерфейсы могут не только принимать информацию от мозга, но и отправлять ее в мозг - например, через сетчатку глаза. Йенс Науманн - слепой, способный «видеть» с помощью нейронного зрительного протеза.[17] Камера улавливает изображение и направляет обработанную версию в зрительную кору головного мозга через электроды. Такой тип взаимодействия – большое преимущество для людей с ограниченными физическими возможностями. Рисунок 4.

Рисунок 4. Нейронный интерфейс.

Таким образом, интерфейсы существуют для того, чтобы люди могли взаимодействовать с нашим миром. Через интерфейс мы можем прояснить, проиллюстрировать, дать возможность, показать взаимосвязь, объединить людей или разделить, управлять ожиданиями и давать доступ к услугам.

1.2. Эволюция графических интерфейсов операционных систем.

Любое устройство, механическое или электронное, в задачи которого входит прямое взаимодействие с пользователем, помимо своей внутренней начинки

должно обладать чем-то, благодаря чему это самое взаимодействие смогло бы осуществляться. Имя этого посредника сегодня известно каждому. Это — интерфейс. Он может быть аналоговым, но обычно под интерфейсом понимают графическую оболочку или иначе GUI операционных систем и программного обеспечения.

Большинству интерфейсов популярных ныне операционных систем свойственно интуитивно-понятное графическое оформление с использованием визуальных эффектов, однако так было не всегда. С точки зрения современного пользователя первые GUI были довольно примитивны, хотя, нужно отдать им должное, это не всегда означало отсутствие качественного по тем временам юзабилити.[18]

Традиционно годом рождения GUI принято считать 1973, именно тогда на свет появился первый в полном смысле этого слова персональный компьютер Xerox Alto, в котором использовался графический интерфейс, но было бы несправедливо при этом не упомянуть о его более ранних предшественниках. В 1962 году учёным Айвеном Сазерлендом была создана программа, которую можно считать первым прообразом графических редакторов.[19] Рисунок 5.

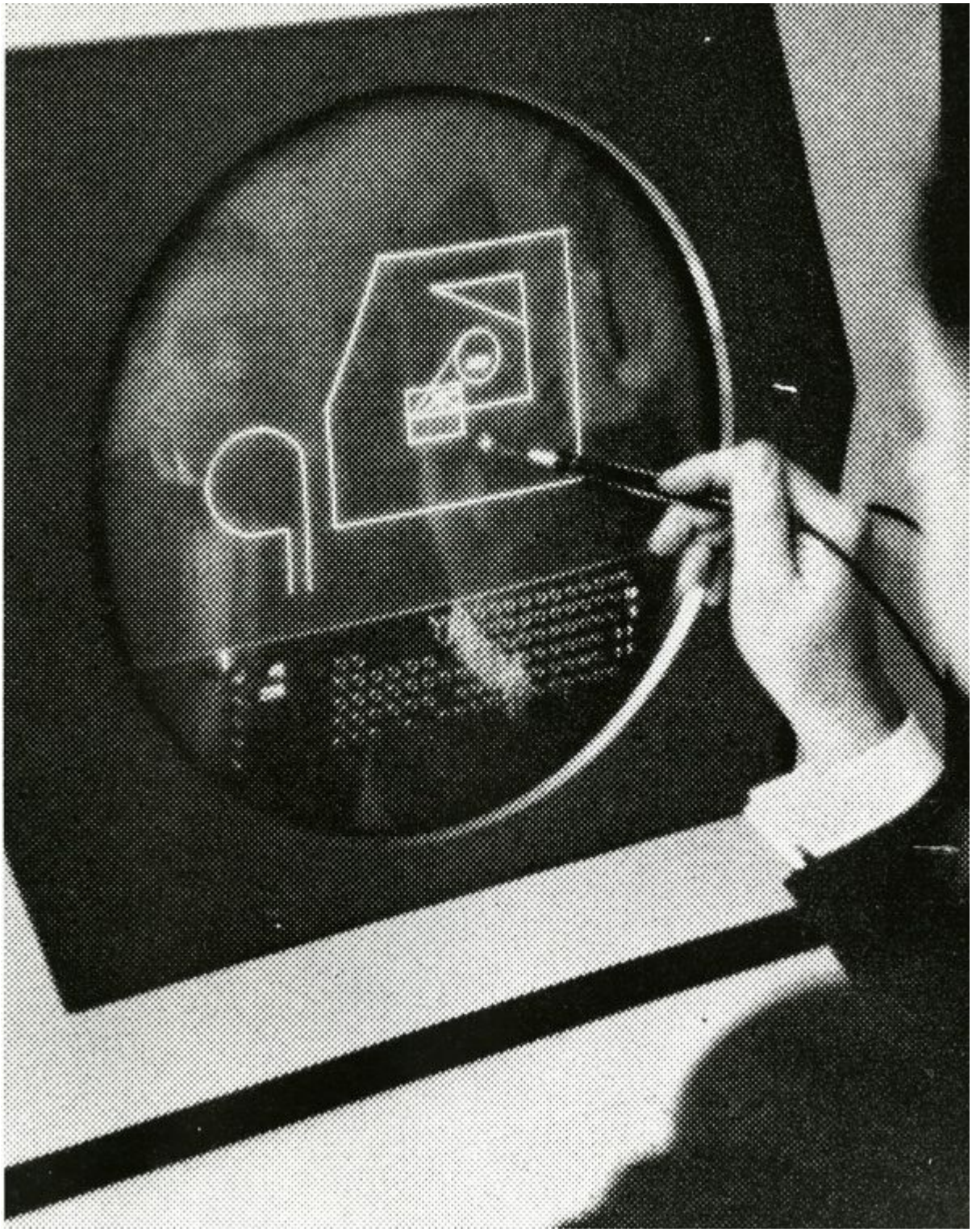


Рисунок 5. Программа Sketchpad

Называлась она Sketchpad и позволяла рисовать на экране фигуры световым пером. Спустя шесть лет учёными Стэндфордского института была представлена первая использующая графический интерфейс компьютерная система oN-Line System, в которой уже тогда был заложен концепт современных окон, мышки и

гипертекстовых ссылок. Но oN-Line System была скорее демонстрацией технических возможностей того времени, оставаясь при этом весьма примитивной. [20]

Родоначальником всех ныне существующих графических интерфейсов правильнее считать GUI, разработанный в рамках проекта Xerox Alto — первого персонального компьютера, созданного в 1973 году. Оболочка Xerox Alto была очень проста, но уже тогда в ней присутствовали меню, кнопки и примитивные окна. Был в ней и курсор мыши с присущими ему функциями выделения, копирования и вставки.[21]

В 1981 году появляется новая система под названием Xerox Star, основанная на той же Xerox Alto, но с более совершенным функционалом и графическим интерфейсом. Рабочий стол Xerox Star, мало чем отличался от нынешних десктопов, если, конечно, не брать в расчёт визуальные эффекты. Рисунок 6.

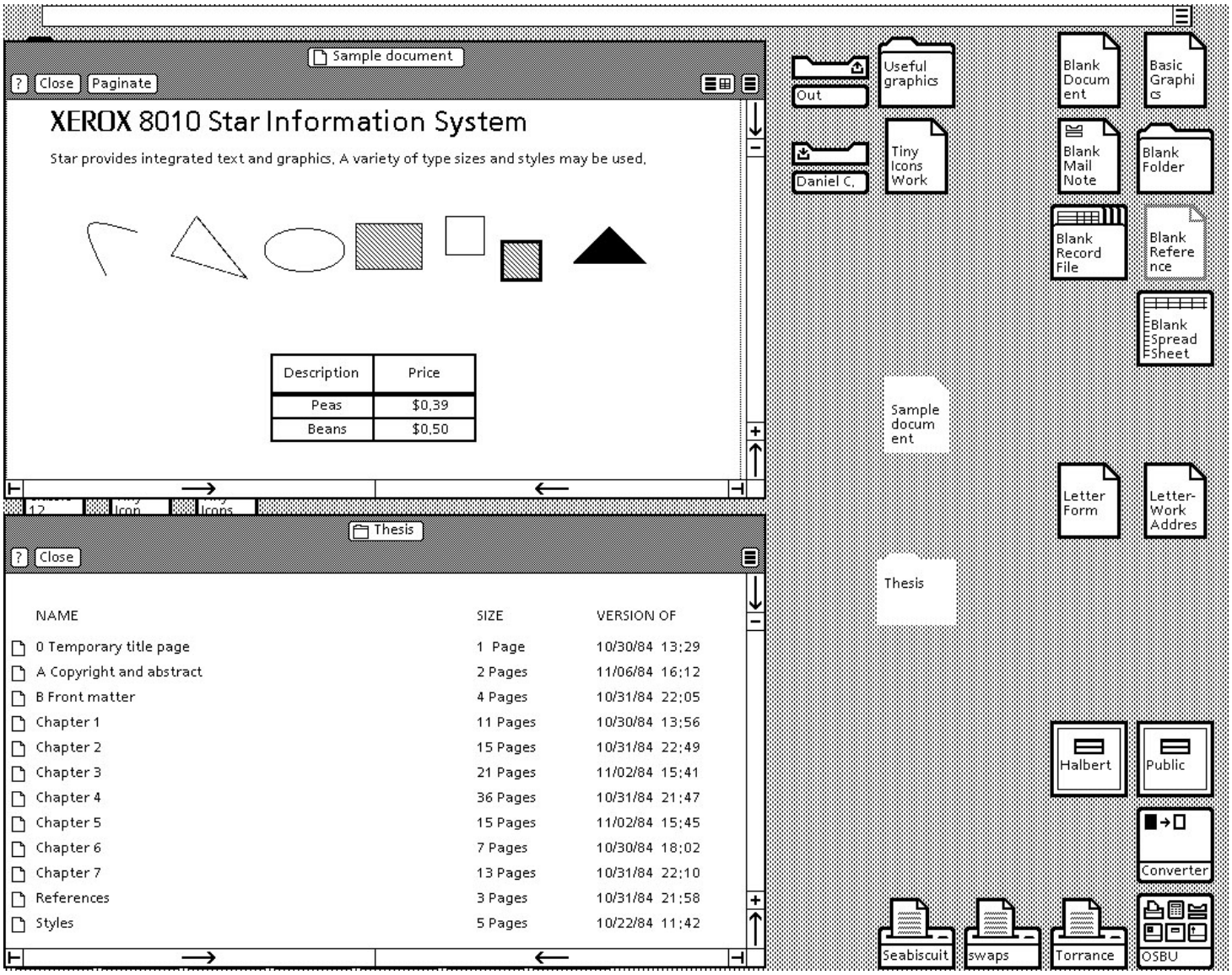


Рисунок 6. Рабочий стол Xerox Star

В его основе лежит тот же принцип использования ярлыков для запуска файлов и перехода по каталогам файловой системы.

Надо отметить, Xerox Star была не единственной на то время операционной системой. В начале 80-х годов свои разработки миру представили компании Apple и Microsoft. Понимая всё значение GUI, но не имея достаточно времени для создания оригинальных оболочек для своих систем, разработчики обеих компаний позаимствовали идеи Xerox Lab, что впоследствии даже привело к конфликту между Стивом Джобсом и Биллом Гейтсом. Джобс обвинил Гейтса в плагиате, что тот, якобы, скопировал интерфейс с Macintosh.[22]

Конечно, Стив не был прав, потому что и он сам, и обвиняемый им Гейтс взяли концепцию GUI у Xerox Lab, просто так получилось, что Джобс оказался первым, и если сравнить интерфейсы Apple Lisa, а также родственного ей Macintosh, то можно увидеть явное сходство с Xerox Star. Если что и было добавлено компанией нового, так это текстовое меню в верхней части рабочего стола, Корзина и ряд ярлыков, чей вид, по мнению «яблочных» дизайнеров, больше соответствовал назначению запускаемых через них программ и функций.

Заимствования идей Xerox Lab, однако, вовсе не означают, что никаких попыток создания оригинальных интерфейсов для операционных систем не предпринималось. В 1986 году программистом Джоном Соча был создан Norton Commander — файловый менеджер для MS-DOS, до этого не имевшей практически никакого графического оформления.[23] Роль окон в нем играли панели, делящие экран по вертикали и содержащие списки папок и файлов. В верхней и нижней части менеджера располагались текстовые меню, позволяющие выполнять те или иные операции. Рисунок 7.

Рисунок 7. Файловый менеджер Norton Commander

Впрочем, GUI в полном смысле этого слова Norton Commander не являлся. Как и вышедшей в 1988 году его аналог DOS Shell, он относится к псевдографическим интерфейсам, имитирующим графику, оставаясь при этом текстовыми. Рисунок 8.

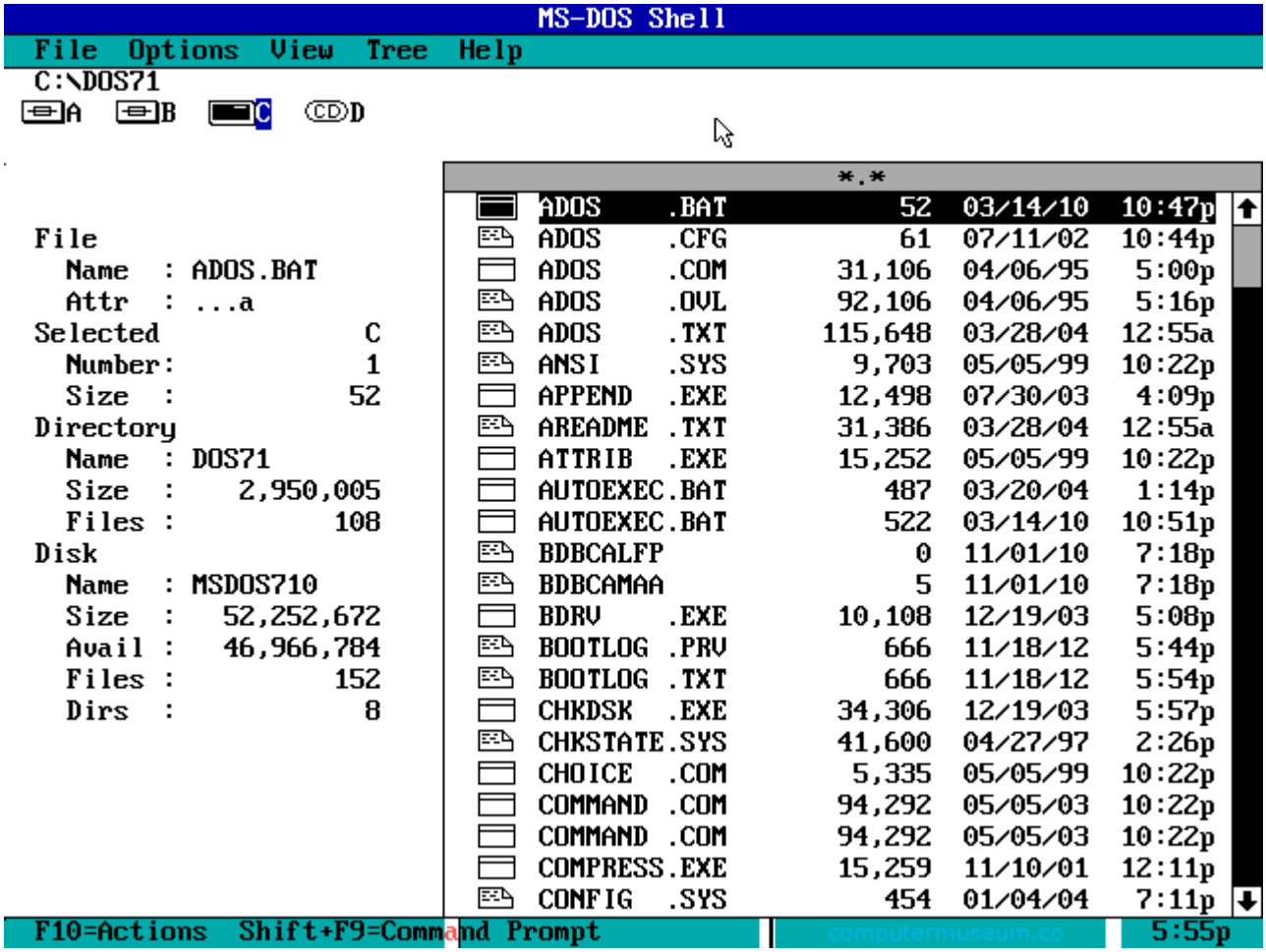


Рисунок 8. DOS Shell

Тем не менее, оба эти приложения существенно облегчили работу с данными, избавив пользователей от необходимости вводить DOS-команды, чем долгое время и обуславливалась популярность этих программ.

Выйдя из команды разработчиков Apple Lisa, в 1982 году Стив Джобс возглавил собственный проект Macintosh**.** Разработанная для маков система получила название Mac OS. Внешне она была похожа на Apple Lisa, но в ней имелись также и только ей одной присущие особенности, причём касались они как внешнего вида элементов интерфейса, так и самого взаимодействия пользователя с оболочкой. Как и Apple Lisa, MacOS 1.1 была основана на оконном принципе, в ней использовались меню, иконки и диалоги.[24]

Оболочка MacOS 1.1 позволяла быстро переименовывать файлы и папки, выделять их, копировать перетаскиванием в место назначения, одновременно закрывать все окна, хотя закрытие окон не всегда предполагало завершения работы приложения, закрывать программы нужно было правильно — через главное меню системы. При

закрытии отредактированных, но не сохранённых файлов появлялось диалоговое окно с запросом на подтверждение сохранения изменений или их отмены.

За семь лет своего существования Mac OS прошла через множество изменений, но почти все они были незначительными и только в седьмой версии появились нововведения, о которых стоило бы упомянуть. Пожалуй, самое главное из них это поддержка цветов, так как до этого интерфейс системы был практически монохромным. Теперь пользователь мог менять цвет иконок папок и некоторых других элементов, делая их синими, желтыми или красными.

Присутствовали в цветовой гамме Mac OS 7.5.5 и другие оттенки. В это же время становится цветным «яблочный» логотип в левой части главного меню. Из прочих изменений можно отметить показ иконок модулей во время загрузки системы, расширение функционала меню, добавление всплывающих подсказок при наведении на доступные в меню опции, а также реализация доступа к приложениям из единой панели управления.[25]

Работа над использованием цвета в графическом интерфейсе была активно продолжена в восьмой версии системы. Системные иконки в Mac OS 8.1 были цветными по умолчанию, а в самой ОС появилось новое приложение Appearance Manager, позволяющее управлять цветовыми схемами. MacOS 8.1обзавелась набором фоновых изображений, кроме того, в качестве фонов пользователь мог устанавливать произвольные картинки.

В этой же редакции впервые появляется знаменитая платиново-серая тема, ставшая впоследствии визитной карточкой всех последующих версий Mac OS. Другим интересным изменением стало применение к иконкам изометрии, благодаря чему они стали походить на трехмерные объекты, не являясь таковыми на самом деле. Были улучшены настройки отображения содержимого файловой системы — файлы стало можно просматривать в виде списков и значков, размер которых также можно было изменять.

Версией 9.2.2 завершается история Mac OS на основе оригинальной операционной системы Macintosh, и казалось, что в ней должно быть больше нововведений, чем в прошлых версиях. В девятой версии действительно много изменений, но коснулись они по большей части функционала, интерфейс же изменился незначительно. Рисунок 9.

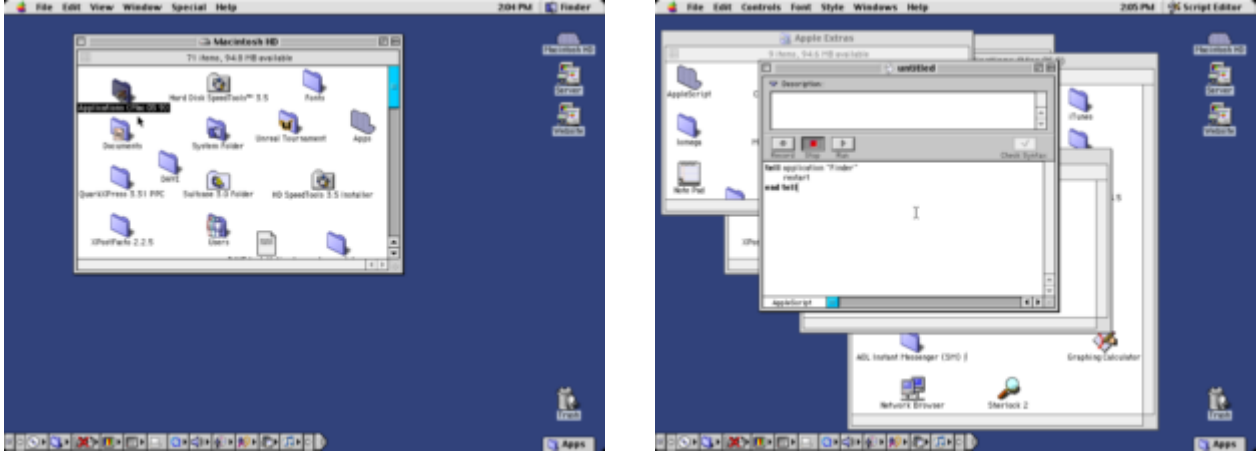


Рисунок 9. Интегрированная поддержка нескольких учетных записей

Из наиболее значимых модификаций, затронувших графическую оболочку, стала интегрированная поддержка нескольких учётных записей. При старте системы на экране появлялось окно выбора профиля, а каждый пользователь мог устанавливать свои темы оформления, причём сторонние графические пакеты тоже поддерживались.[26] В этой же версии также была улучшена панель управления звуком и добавлена возможность установки голосового пароля.

Покинув Apple, в 1985 году Стив Джобс основал собственную компанию NeXT Computers, разработчиками которой была создана ОС NeXTSTEP. Основой новой системы стало ядро Mach**,** а идеи графического оформления позаимствованы у Mac OS и более ранней Apple Lisa.

Подобно Mac OS, первоначально NeXTSTEP имела упрощенный монохромный интерфейс, основанный на оконном принципе с использованием меню, иконок и диалоговых окон. Полная поддержка цветов появилась только в версии 3.3, ставшей последней. От поздних версий Mac OS система NeXTSTEP 3.3отличалась более чистым и лаконичным оформлением.

В 1985 году компания Microsoft представляет свою графическую оболочку для MS-DOS с говорящим названием Windows**.** Оболочка частично поддерживала цветную графику, в ней имелись 32×32-пиксельные иконки, простые меню и диалоги. Фиксированной области, в которой бы отображались значки, запущенных приложений пока не было, располагаться они могли в любом месте экрана, перекрываясь при этом открытыми окнами.

Сами окна в первой версии были довольно примитивными. Их можно было перетаскивать мышкой, изменять их размер, но при этом сами они не могли

перекрывать друг друга. Сворачивать их также было нельзя. Интерфейс Windows 1.0 облегчал работу с системой и файлами, избавив пользователя от необходимости вводить команды в консоли, но в то же время ему недоставало удобства. Так что в плане юзабилити первая версия Windows значительно уступала системам от Apple.

Версии Windows 1.0, 2.0 и 3.0 не были операционными системами в том смысле слова, в котором его принято понимать сегодня. Это были скорее графические оболочки MS-DOS, первые признаки, выделяющие Windows в отдельную ОС, появились только с выходом версий 3.1 и 3.11, но относятся они не столько к GUI, сколько к функционалу. В плане графического оформления существенных изменений было не так уже и много.

В Windows 3.11 уже имеется полная поддержка цветов, окна могут перекрывать друг друга, их можно сворачивать и разворачивать. Незначительно улучшается графика отдельных элементов (объемные кнопки и полосы прокрутки), используются пропорциональные шрифты, внешний вид программ File Manager **и** Program Manager реализуется в стиле самой оболочки. Цвета элементов интерфейса пользователь может менять по своему усмотрению.

Настоящая революция в оформлении Windows свершилась в 1995 году, именно тогда в системе появляются хорошо ныне всем знакомые кнопка «Пуск**,** Проводник **,** Панель задач и Рабочий стол со значками», который в тоже время являлся отдельной папкой. В этой же версии был реализован показ дисков в папке «Мой компьютер» и способ управления файлами из меню, вызываемого правой кнопкой мыши. Немаловажным нововведением стал переход на 32-битную архитектуру.

Интересна также история Windows 95 и та роль, которую она сыграла в крахе проекта OS/2 — операционной системы совместного детища Microsoft и IBM. На момент выхода Windows 3.0 между компаниями возникли разногласия. Microsoft стремилась продвигать Windows, а IBM ставило приоритетом разработку OS/2. В итоге между компаниями был заключен договор, согласно которому IBM должна была заниматься OS/2 2.0 и Windows 3.0, а Microsoft — OS/2 3.0.

Однако глава Microsoft решил поступить по-своему, объявив OS/2 2.0 устаревшей, а более новую OS/2 3.0 переименовав в Windows NT. В это же время Microsoft выпускает Windows 3.1, а затем и обновление для версии 3.1 под кодовым названием Chicago, положенное в основу будущей Windows 95. После этого пути IBM и Microsoft разошлись окончательно. Некоторое время IBM ещё занималась

разработкой OS/2, но выход более конкурентоспособной Windows 95 окончательно добил её, и IBM вынуждена была свернуть проект.[27]

Поскольку мы затронули тему конфликта между IBM и Microsoft, было бы несправедливо обойти вниманием OS/2, плод некогда плодотворного сотрудничества обеих компаний. После ссоры и окончательного разрыва с Microsoft, разработчики IBM продолжили работу над OS/2. В вышедшей в 1996 году версии OS/2 Warp 4 произведены значительные улучшения внешнего оформления рабочего стола и графической оболочки в целом. На рабочем столе имелись иконки, обеспечивающие доступ к разным системным настройкам, но они же могли играть роль каталогов, в которые пользователь мог помещать файлы и папки, присутствовала Windows-подобная панель задач и единое меню доступа ко всем

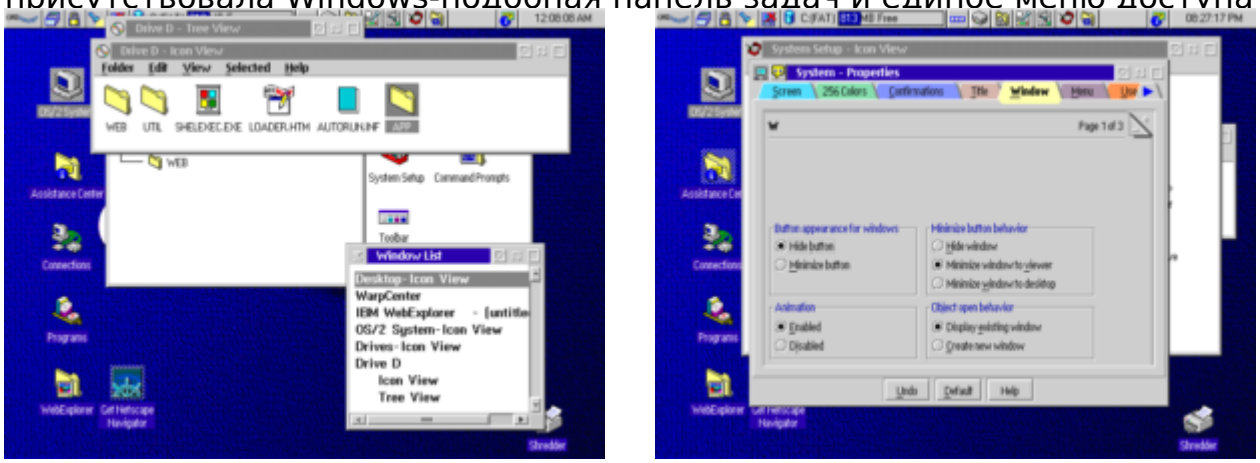


Рисунок 10. Версия OS/2 Warp 4

OS/2 Warp 4 имеет много общего с Windows, но есть в ней и весьма существенные отличия. Перетаскивание по умолчанию в OS/2 Warp 4 осуществлялось не левой, а правой кнопкой мыши, «Корзина» служила не для временного хранения удаленных файлов, а для немедленного их уничтожения. Диалоговые окна в этой системе не имели кнопок «Сохранить» или «OK», данные сохранялись автоматически при закрытии окна, если только пользователь не отменял действие. Другой интересной особенностью системы была возможность просмотра древовидной структуры диска в окнах.

Выход Windows в 1985 году не ослабил интерес к MS DOS, которая по-прежнему пользовалась популярностью, равно как и созданные под неё графические оболочки, среди которых выпущенная фирмой Tandy среда DeskMate заслуживает особого внимания. DeskMate — это не просто графическая надстройка, это

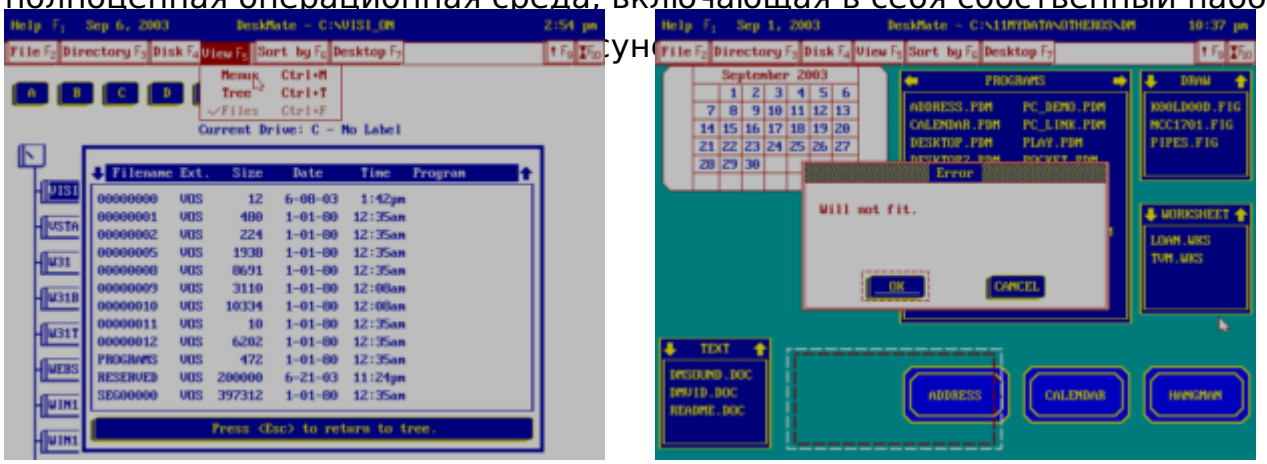
полноценная операционная среда, включающая в себя собственный набор


Рисунок 11. Выпущенная фирмой Tandy **среда** DeskMate

В отличие от Norton Commander, в DeskMate 3.05 имелись полноценные меню, кнопки и некое подобие окон, которые можно размещать на примитивном рабочем столе. В среду был интегрирован файловый менеджер с ограниченной поддержкой древовидной структуры, встроенный учебник, календарь, органайзер, СУБД, текстовый и векторный графический редакторы. Другой отличительной чертой DeskMate 3.05 являлась поддержка основных цветов.

AmigaOS — операционная система, специально созданная для компьютеров семейства Amiga в 1985году. Интерфейс первых версий представлял собой нечто среднее между оболочкой Apple Lisa и псевдографическими оболочками DOS, но уже в AmigaOS 3.5 появились перекрывающие друг друга окна, цветные иконки для быстрого доступа к файлам, приложениям и дисковым накопителям.[29]

Если брать в целом, AmigaOS 3.5 имеет много общего с MacOS. Меню в верхней части экрана показывает опции в зависимости от того, какие приложение являются на данный момент активными, окна оснащены простейшими элементами управления, есть полосы прокрутки. Отличительной чертой AmigaOS является функция, позволяющая работать с несколькими экранами, причём каждый экран мог иметь свое разрешение и глубину цвета.

RISC OS 4. - маленькая, быстрая и несколько необычная Unix-подобная операционная система, разработанная для платформы Raspberry Pi. Внешне отличалась минималистичным пользовательским интерфейсом, поддержкой всех доступных на тот момент разрешений для компьютеров Acorn, перетаскивания, в

ней имелись окна, своеобразная панель задач в нижней части рабочего стола, цветные иконки и псевдотрёхмерные декоративные элементы управления.

Собственными фишками RISC OS 4 являлись однопользовательская многозадачность, модульность и двоичный интерфейс приложений, все файлы в ней представляли собой тома, приложения также были реализованы в виде каталогов с восклицательным знаком. Несмотря на минималистичность, с точки зрения пользователей, привыкших к окружению Windows или LXDE, рабочий стол RISC OS 4 мог показаться неудобным, так как многие способы управления в нём имели существенные отличия.

BeOS 5.0 PE - достаточно мощная операционная система, созданная компанией Be Inc. и ориентированная на работу с мультимедиа. Изначально разрабатывалась для компьютеров BeBox, но затем перешла на Macintosh, а потом уже и на PC. История BeOS тесно связана с Apple, так как основателем Be Inc. являлся никто иной, как Жан-Луи Гассье — бывший исполнительный директор Apple. Помимо многопоточности и поддержки многопроцессорных архитектур, главной примечательностью BeOS 5.0 PE был её интерфейс, обладающий чертами Windows и систем от Apple.

Оболочка системы выгодно отличалась хорошо продуманным юзабилити, и соединённой с изяществом простотой. Как и положено, в ней имелись способные перекрывать друг друга окна, меню, «сборная»панель задач, имеющая сходство с аналогичным элементом оболочки Windows, полноцветные иконки и папки, которые можно было размещать на выполненном в минималистском стиле рабочем столе. Отличительной чертой BeOS 5.0 PE являлось строение окон — вместо традиционного заголовка в них использовались расположенные поверх окна вкладки.

QNX 6.2.1 — малоизвестная канадская Unix-подобная система, относящаяся к типу так называемых операционных систем реального времени. Будучи универсальной, она отличалась высокой скоростью работы и нетребовательностью к аппаратным ресурсам.[30] Графическая оболочка системы называется Photon, но её рабочий стол имеет много общего с десктопом Windows XP. В нижней части экрана имеется горизонтальная панель управления с подобием кнопки Пуск и вертикальная панель управления справа, поддерживается смена фоновых изображений. Рисунок 11.

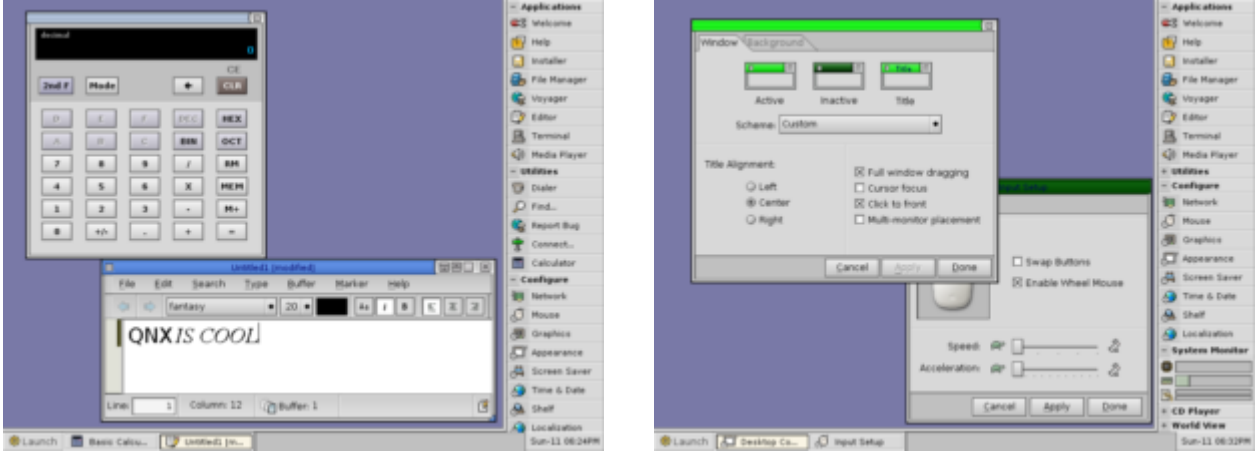


Рисунок 11. Графическая оболочка системы Photon.

Окна QNX 6.2.1, в которых запускаются приложения, имеют аналогичное строение с окнами Windows. Их можно сворачивать и разворачивать, масштабировать, располагать каскадом и прочее. Проводника, как его принято понимать в Windows *в* QNX 6.2.1 нет, его заменяет файловый менеджер QNX Photon, сходный по функциональности с Проводником Windows. Благодаря своей лёгкости, удобству и простоте QNX 6.2.1 в своё время конкурировала и в чём-то даже опережала Windows, однако приложений под QNX писалось мало, что и определило её дальнейшую судьбу. Сегодня QNX используется в основном на специализированных устройствах.

Системы Linux, особенно ранние, обычно воспринимались как ориентированные на компьютерных гиков, однако нельзя сказать, чтобы разработчики Unix-подобных систем ничего не предпринимали для того, чтобы сделать свои продукты столь же удобными, как и Windows. Примером тому может послужить Red Hat 8.0 — основанная на Linux Fedora ОС с оболочкой GNOME 2.06, кстати, изначально разрабатываемой именно для Red Hat и только потом распространившейся и на другие дистрибутивы Linux.[31] Рисунок 12.

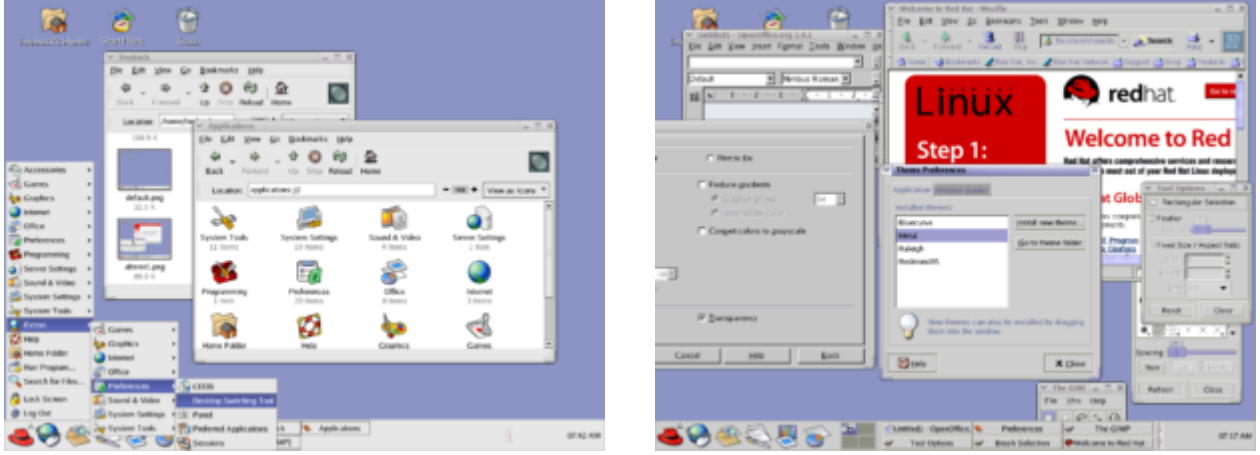


Рисунок 12. Red Hat 8.0 **с оболочкой** GNOME 2.06

Red Hat 8.0 имеет простой и чистый рабочий стол с возможностью размещения на нём иконок и прочих объектов, Windows-подобную панель задач, окна, элементы управления которых располагались не в левом, как это свойственно многим Linux-системам, а в правом верхнем углу, есть в Red Hat 8.0 даже своя кнопка и меню Пуск. Сами меню организованы по каскадному принципу, что ещё больше в плане оформления роднило эту систему с Windows.

Анализ теоретической главы выявил следующее:

Во- первых, интерфейсы – это не памятники самим себе. Интерфейсы выполняют конкретные задачи и их эффективность измерима. Однако они могут выходить за рамки чисто практического применения. Лучшие интерфейсы те, которые вдохновляют, пробуждают чувства, удивляют и усиливают наш опыт общения с миром.

Во-вторых, за более чем полвека своего существования ЭВМ проделали огромный путь, развившись в сложные и мощные системы, нашедшие применение практически во всех современных отраслях. Сравнивать первые электронные вычислительные машины с современными суперкомпьютерами это всё равно, что сравнивать Монгольфьер с космическим кораблем. Но тем более удивительным кажется то, что их интерфейсы не претерпели кардинальных изменений, если не считать перехода собственно к графическим оболочкам.

Если присмотреться к интерфейсам современных операционных систем, то можно заметить явные сходства с графическими оболочками родоначальников всех GUI Xerox Alto и Apple Lisa. Это – не нехватка воображения дизайнеров или некая необходимость, продиктованная рамками физиологии пользователей. Да,

первые графические интерфейсы были примитивны, но была в них одновременно и та подкупающая простота, которой так иногда не хватает перегруженным визуальными эффектами оболочкам современных программ и операционных систем.

2. ПРОЕКТИРОВАНИЕ ФРЕЙМВОРКА

2.1. Понятие Фреймворка

Фреймворк – это программная платформа, которая определяет структуру какой – либо системы; программное обеспечение, которое облегчает разработчикам создание и объединение различных компонентов большого программного проекта. Разработка и использование Фреймворков является неким каркасным подходом, в котором какая – либо конфигурация программы состоит из двух частей: первая – постоянная часть является каркасом, не имеющимся в любых конфигурациях и несущим в себе гнезда, где размещается вторая, переменная часть – сменные модули или точки расширения.[32]

Понятие Фреймворка иногда путают с понятием библиотеки. На самом деле эти понятия различны. Библиотека в отличие от Фреймворка может быть представлена в программном проекте в качестве набора подпрограммного проекта и не накладывает на нее никаких ограничений. Фреймворк же в свою очередь задает правила для построения архитектуры приложения, диктуя поведение по умолчанию уже на начальном этапе разработки. Он является каркасом, который нужно будет расширять и изменять согласно указанным требованиям. В качестве примера Фреймворка можно привести такие программные продукты как NET Framework или Entity Framework, а примером библиотеки можно считать модуль электронной почты. Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, Фреймворк может содержать в себе большое число разных по тематике библиотек.[33]

Еще одним из ключевых отличий Фреймворка от библиотеки, считается инверсия управления: пользовательский код вызывает функции библиотеки (или классы) и получает управление после вызова. Во Фреймворке пользовательский код может реализовать конкретное поведение, встраиваемое в общий, абстрактный код Фреймворка. При этом Фреймворк вызывает функции (классы) пользовательского

кода. Обычно Фреймворк определяется множеством каких – либо конкретных и абстрактных классов и определениями способов для их взаимодействия между собой.[34] Конкретные классы реализуют взаимные отношения между классами. Абстрактные классы обычно являются некими точками расширения, где используются и адаптируются каркасы. Точка расширения – это та часть Фреймворка, для которой не приведена реализация. Процесс создания Фреймворка состоит в выборе подмножества задач какой – либо проблемы, а также их реализаций. В процессе реализаций общие средства решения заключаются в конкретных классах, а изменяемые средства выносятся в точки расширения.

2.2. Инструментальные средства проектирования

Исторически сложилось, что все браузеры, мобильные и десктопные, на всех платформах понимают всего лишь три вещи: HTML, CSS и JavaScript (не путать с Java.).Были попытки подружить браузеры с другими технологиями: Visual Basic, Java, ActiveX, — но все они провалились, потому что производители железа и браузеров не смогли договориться об открытых стандартах. Остались только открытые стандарты, разрабатываемые публичными консорциумами и рабочими группами. Например, W3C разрабатывает HTML и CSS.

Итак, у нас есть три технологии:

- HTML отвечает за структуру страницы.
- CSS — за ее оформление (визуальное и адаптив для разных экранов).
- JavaScript — за взаимодействие страницы с пользователем (по изначальной задумке; сейчас-то уже практически вообще за все).

Каждая технология развивается независимо, у каждой есть несколько версий. Самые свежие на сегодня версии: HTML5,CSS3, ECMAS cript 2018 (это стандарт Java Script). Браузеры тоже развиваются независимо. Кто-то (то Internet Explorer, то Safari) отстает от стандартов, кто-то (обычно Chrome или Firefox) впереди и внедряет экспериментальные фичи.

Отсюда постоянная головная боль фронтенд - разработчиков: сайт должен выглядеть во всех браузерах одинаково (причем именно так, как его придумал дизайнер). Плюс работать быстро, безопасно и в соответствии со спецификацией.

Вдобавок HTML и CSS — это не языки программирования, а языки разметки: один лишь синтаксис, набор команд — конструкций и правил — для представления содержимого страницы. Ну, к примеру, в них нет как таковых классов, объектов, функций, методов, присущих языкам программирования. Чтобы наделить веб - сайт функциональностью и бизнес - логикой, приходится подключать язык программирования на стороне сервера или на стороне клиента (в браузере), а чаще всего — и там и там.

Стандартный Javascript — не самый эффективный и приятный для работы язык программирования. Специалисты критикуют его за то, что даже для самых простых операций приходится писать очень много строчек кода, постоянно повторяться. Это замедляет разработку. Потому верстальщики и программисты ищут способы использовать продвинутые инструменты вместо «чистых» JavaScript, HTML и CSS.

Чтобы проиллюстрировать пример неэффективности всей этой связки, возьмем такую простую конструкцию, как таблица. В HTML есть набор тегов для создания таблицы (основные из них — table, th, tr, td). С их помощью можно создать только сетку таблицы с минимальными настройками внешнего вида: задать ширину колонок, размеры ячеек и в принципе всё. Добавив CSS, можно (изрядно помучавшись) придать ей пристойный вид и при должном старании адаптировать для разных экранов. Но мы все равно не сможем сортировать таблицу по одной или нескольким колонкам, показать порядок сортировки, добавлять или удалять колонки и столбцы, перетаскивать данные из одной ячейки в другую, использовать формулы для подсчета сумм по строкам и столбцам, применить условное форматирование ячеек (подсветить отрицательные, например) и т. п.

Всё это на HTML и CSS невозможно сделать, потому что в HTML нет такого объекта, как таблица, и нет методов работы с ней, которые поддерживались бы любым браузером. Разработчики стандарта 20 лет назад не предполагали, что пользователи захотят работать с содержимым веб-страницы. Похожие проблемы у нас будут, если мы захотим отправить на сайт пачку файлов (например, добавить несколько вложений к письму в веб-почте), выбрать интервал времени (например, запланировать встречу в календаре) или представлять одну и ту же информацию разными способами (например, показывать товары в Интернет -магазине по желанию пользователя карточками или строками). Для всего этого в HTML и CSS нет подходящих решений, потому на помощь приходит JavaScript — язык программирования, который может манипулировать объектами в структуре HTML и применять к ним стили CSS.

Каждый раз писать код на JavaScript, чтобы сделать ту же сортировку таблиц, — это, разумеется, не легкая задача. И появились библиотеки — наборы готовых функций на JavaScript, выполняющие типовые операции с HTML-кодом страницы. Пример такой библиотеки — jQuery. Этих библиотек за двадцать с лишним лет существования JavaScript появилось великое множество. Программистам приходилось комбинировать библиотеки, дружить их между собой, обновлять (ведь каждая развивается своим чередом), следить за совместимостью. Да еще самим код писать — не все же есть в библиотеках. Подход с библиотеками до сих пор живет. В небольших проектах достаточно подключить одну - две библиотеки для конкретных улучшений. Например, чтобы рисовать красивые графики, подключаем бесплатный Chart.js или платный AmCharts. Если нужна анимация и отзывчивость интерфейса — тот же jQuery, для работы с элементами интерфейса есть смысл взглянуть на Sencha Ext JS и т. п.

Для HTML+CSS тоже стали появляться подобные «полуфабрикаты» — заготовки из кусков кода, которые решают типовые проблемы верстки. Например, многоколоночная верстка, закрепленный на странице хедер или футер и прочие типовые задачи, которые выгоднее решить один раз, а потом применять в новых проектах.

Так появились первые фронтенд - каркасы разработки, или Фреймворки. Почему они не библиотеки? Потому что это не набор готовых функций, которые можно добавить к проекту и использовать точечно на отдельных страницах. Каркас (Фреймворк) предполагает, что весь проект будет следовать заданной им структуре.[35] То есть он задает ограничения, которых нужно придерживаться, чтобы ускорить разработку, точнее следовать стандартам, снизить порог вхождения разработчиков в проект и т. п.

Самый известный образец HTML+CSS - Фреймворка — Bootstrap (вот примеры, вот один из компонентов — карусель, вот другой — кнопки). Важно, что все сделано на стандартных HTML и CSS и будет работать (и работать более или менее одинаково) во всех браузерах. Фреймворки уже содержат подогнанные друг к другу совместимые библиотеки, так что разработчику не нужно ничего обновлять, помнить про ограничения и заботиться о совместимости. Так, многие компоненты Bootstrap содержат код на JavaScript с использованием библиотеки jQuery.

Другой известный Фреймворк, Foundation, кроме jQuery использует библиотеки Modernizr и FastClick. Два Фреймворка в приложении будут

конкурировать за базовые вещи. К примеру, один захочет 12-колоночную сетку, другой — 16-колоночную; они могут использовать одинаковые названия методов JavaScript для разных целей и т. д. Поэтому между собой Фреймворки не совместимы: нужно определиться и выбрать один. Если у вас проект на Bootstrap, а вам нужны вот такие вот звездочки из Foundation — то «поженить» их не получится. Главная проблема современного веба — разрозненность технологического стека. Невозможно выучить все технологии, знать и уметь их правильно готовить.

Пять лет назад появилась вполне здравая идея — поскольку большинство сайтов и мобильных приложений оперируют ограниченным набором шаблонов, разумно отделить представление данных от собственно данных.[36] Пусть бэкенд занимается только хранением, обработкой и безопасностью данных (извлекает их из хранилища, проверяет наличие прав доступа, передает их на Фронтенд), а всё остальное поручим клиенту (браузеру). Дадим ему пачку данных и шаблон — набор инструкций по превращению данных в верстку. Фреймворк на клиенте будет подставлять данные в шаблон, реализовывать бизнес-логику и вообще манипулировать страницей, наводить красоту и т. п.

Такие Фреймворки называются реактивными, потому что в них состояние интерфейса автоматически реагирует на изменение данных. Если сказать «вот эта переменная управляет цветом вон той кнопки», а потом поменять переменную — цвет кнопки изменится сам, перекрашивать ее вручную не надо. Реактивное программирование — вариант многопоточного, при котором вместо системы «запрос — ожидание ответа — получение ответа — обработка» работает принцип «запрос (послали и забыли) — ответ — обработка ответа».[37]
Примеры: React, Angular, Vue и еще десятки менее известных. Итак, на бэкенде можно использовать любой язык программирования, добывать им данные, упаковывать их в JSON, XML или что угодно другое машинно-читаемое и отдавать на фронт. А на фронтенде JS-Фреймворк делает всю чистовую работу: рисует контролы, анимирует их, проверяет данные, представляет их в соответствии с локальными настройками пользователя и реализует бизнес-логику.

Что касается надстроек: JS - Фреймворки сразу же обросли библиотеками GUI-компонентов, или надстройками, которые решают конкретные вопросы отзывчивого и богатого GUI в рамках конкретного Фреймворка. Примеров множество: KendoUI — набор UI-компонент для jQuery, React, Angular и Vue, ReactStrap — Bootstrap для React,ReactNative — GUI Фреймворк от "Фейсбука" и т. д. Некоторые надстройки реализованы для нескольких Фреймворков, Onsen,

например, или тот же KendoUI.

2.3. Проектирование Фреймворка на примере приложения

Как известно, большинство популярных Фреймворков создавалось не с пустого места. Большинство компаний, имеющих в своем арсенале какие – либо собственные Фреймворки, получили их после разработки крупных приложений. То есть, они сдавали разработанные проекты заказчику и, впоследствии, оставляли себе каркас приложения для разработки будущих проектов. Полученный шаблон и является Фреймворком.

В данном случае, Фреймворк для построения распределенной информационной системы получился в процессе разработки некого приложения Bookstore. Оно представляет собой автоматизированную систему по продаже книг. Также и описание Фреймворка будет вестись на

примере этого приложения.

Так как в качестве среды разработки выбрана платформа .NET Framework, именование всех проектов, классов, интерфейсов, методов и других различных структур будет вестись в соответствии с общими соглашениями об именовании .NET. Данный подход упростит процесс ознакомления и изучения для будущих разработчиков, собирающихся использовать Фреймворк для построения распределенной ИС. Структура приложения Bookstore, на примере которого происходит описание Фреймворка, представлена на Рисунке 13.

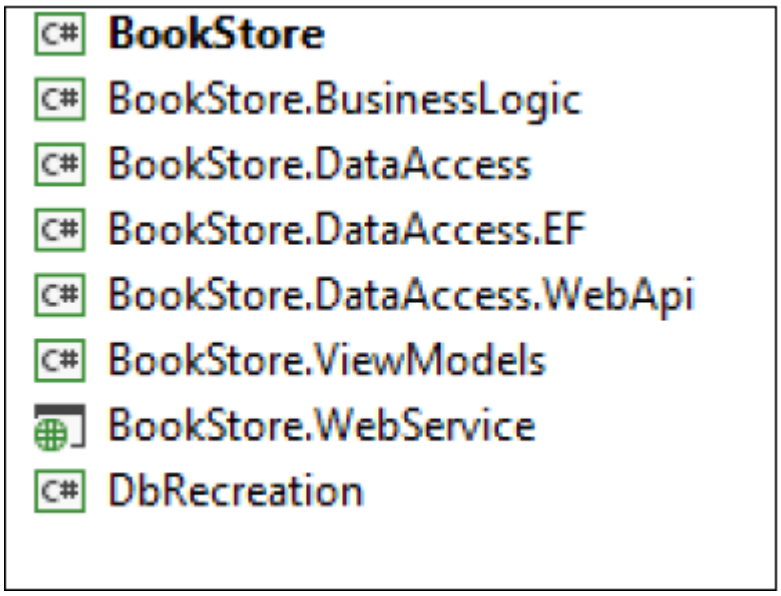


Рисунок 13. Структура приложения на Bookstore

2.4. Проектирование базы данных.

Как уже упоминалось, первичной технологией доступа к данным послужил Entity Framework, а системой управления базами данных – Microsoft SQL Server. В качестве главного подход был выбран Code First. С помощью данного подхода, разработчику не обязательно создавать базу данных вручную, а она проектируется из модели автоматически при написании кода. Для генерации БД, требуется указать строку подключения с такими параметрами:

1) имя подключения;

2) строка подключения, где указывается имя сервера БД, имя базы

данных и другие необязательные параметры;

3) имя провайдера.

Строка подключения на примере приложения Bookstore представлена на Рисунке 14.

```
<connectionStrings>
  <add name="BookStoreDbContext"
       connectionString="Server=ДЕНИС-TOSH\SQLEXPRESS;
                         Database=BookStore;
                         Integrated Security=True;"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

Рисунок 14. Строка подключения приложения Bookstore

Ключевой частью доступа к данным является проект Data Access.EF. Его структура представлена на рисунке 15. В нем располагается класс, представляющий из себя контекст базы данных, который описывает основные свойства модели. Также в данном проекте находится папка Migrations, в которой расположены миграции, создаваемые автоматически при изменении каких-либо свойств в модели. Миграции создаются при помощи команды enable - migrations в менеджере консоли. После выполнения этой команды, в проекте создается папка миграций с файлом конфигурации. Этот файл представляет собой объявление одноименного класса, который устанавливает настройки конфигурации. После этого, также с помощью менеджера консоли, требуется создать сами миграции. Это происходит при вызове команды Add-Migration. Затем, после автоматического создания миграций, разработчику требуется только обновить базу данных при помощи команды Update-Database в том же менеджере консоли. Миграции во многом могут упростить жизнь разработчику, поскольку ему не приходится пересоздавать базу данных по – новому при каком – либо изменении в модели. Рисунок 15.

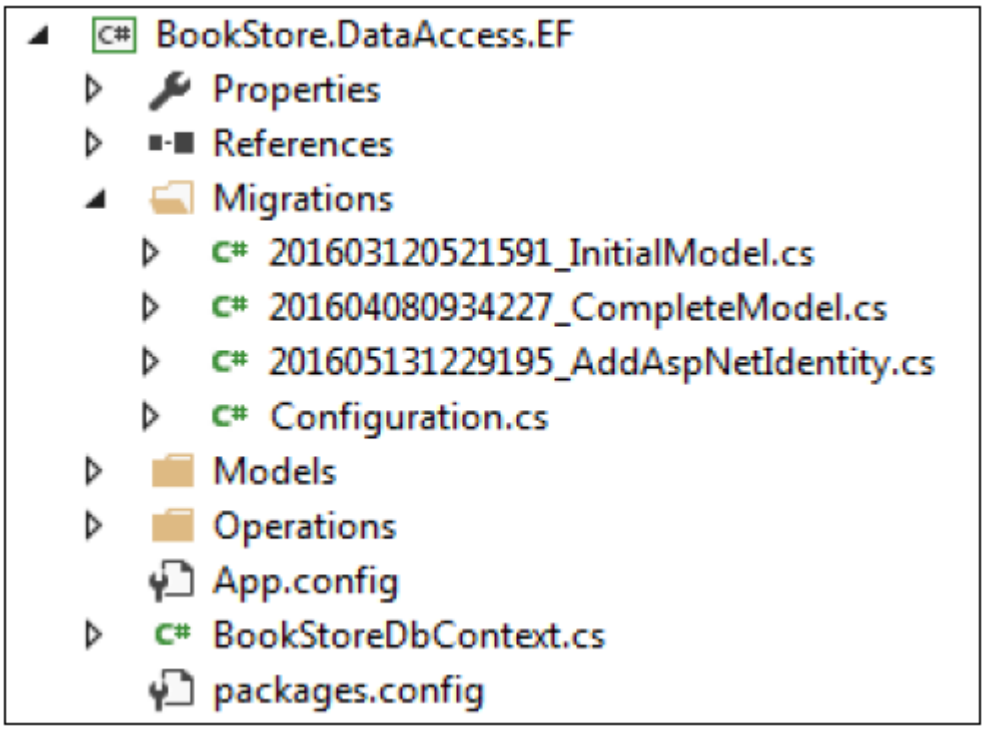


Рисунок 15. Структура проекта Data Access.EF.

Для постоянной перегенерации базы данных был разработан проект, в виде консольного приложения, названный DbRecreation, структура которого представлена на рисунке 16.

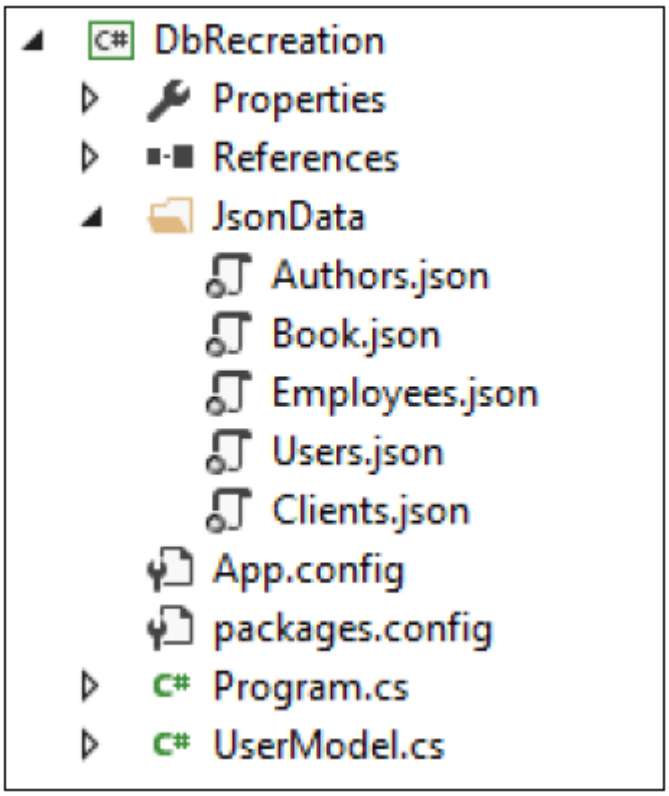


Рисунок 16. Структура проекта DbRecreation

В этом проекте находится папка JsonData, в которой разработчики могут располагать файлы формата JSON, с данными, которые будут заноситься в БД. Также разработчики могут наполнять базу данных различными тестовыми данными, непосредственно в классе Program. Диаграмма этого класса представлена на рисунке 17.

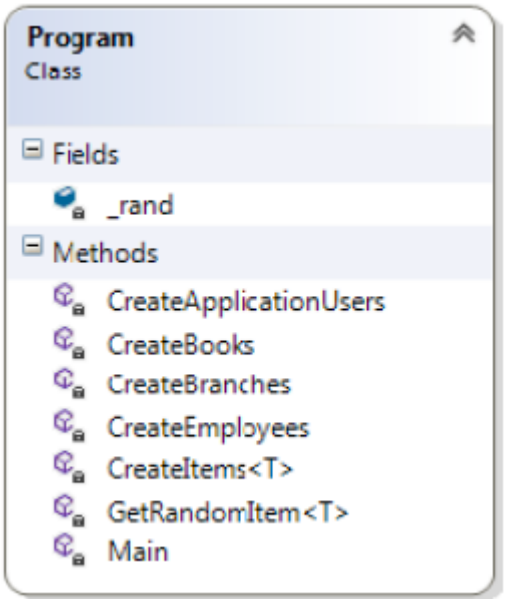


Рисунок 17. Класс Program проекта DbRecreation

В главном методе Main этого класса, происходит проверка на существование базы данных, указанной в строке подключения. Если она существует, то происходит процесс ее удаления. Затем инициализируется новая база данных на основе миграций и происходит вызов методов, создающих объекты моделей. Например, для приложения Bookstore, вызываются методы, создающие книги, авторов книг, книжные подразделения и т.д. Также присутствует общий метод, именованный CreateItems, который непосредственно является частью фреймворка. Этот метод считывает данные, расположенные в папке JsonData, конвертирует их и отправляет в базу данных. На примере приложения Bookstore, в базе данных было создано 15 таблиц, содержащих данные о книгах, авторах, клиентах, пользователях и т.д. На рисунках 18 и 19 представлена схема получившейся базы данных.

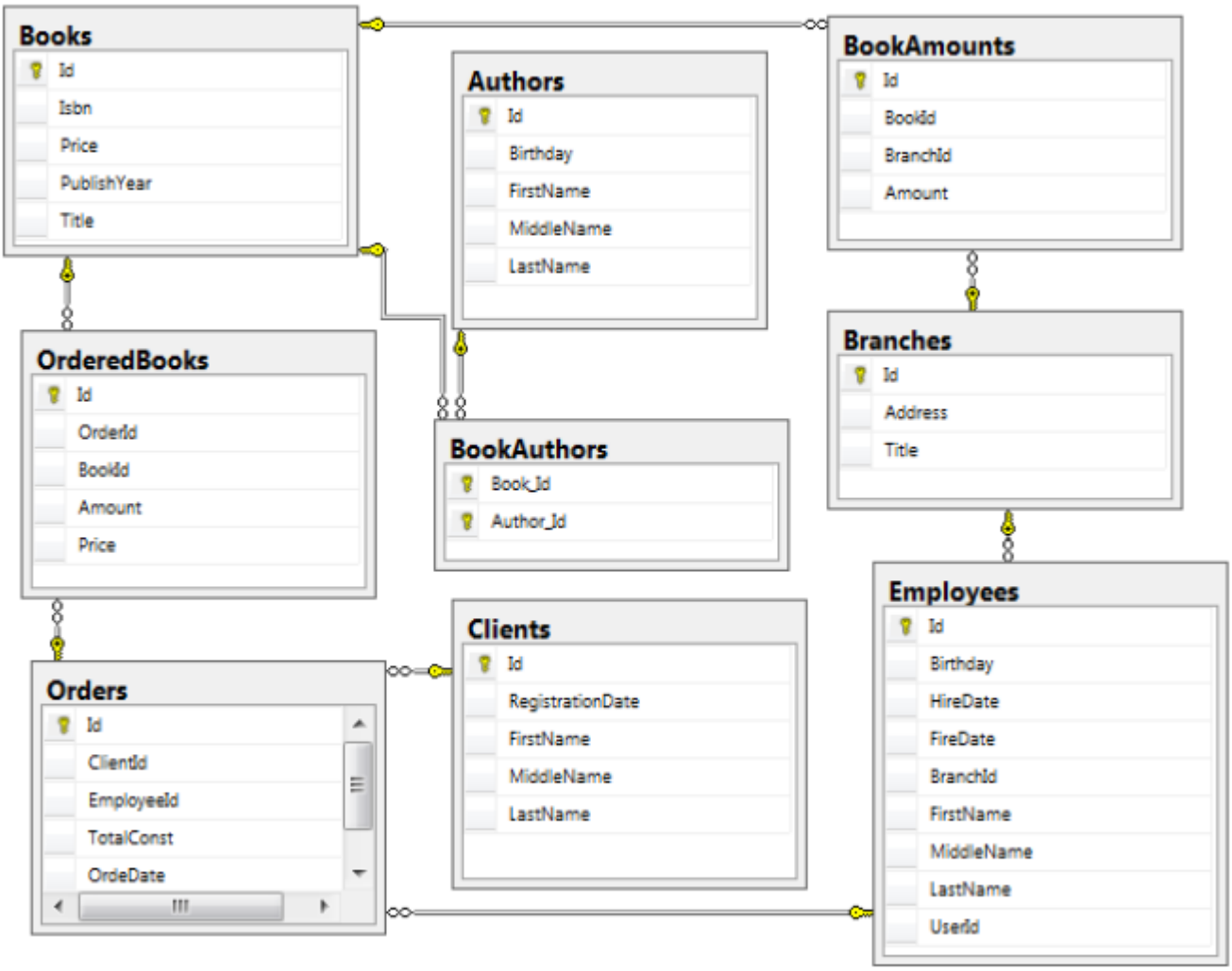


Рисунок 18. Схема базы данных приложения Bookstore. Часть 1

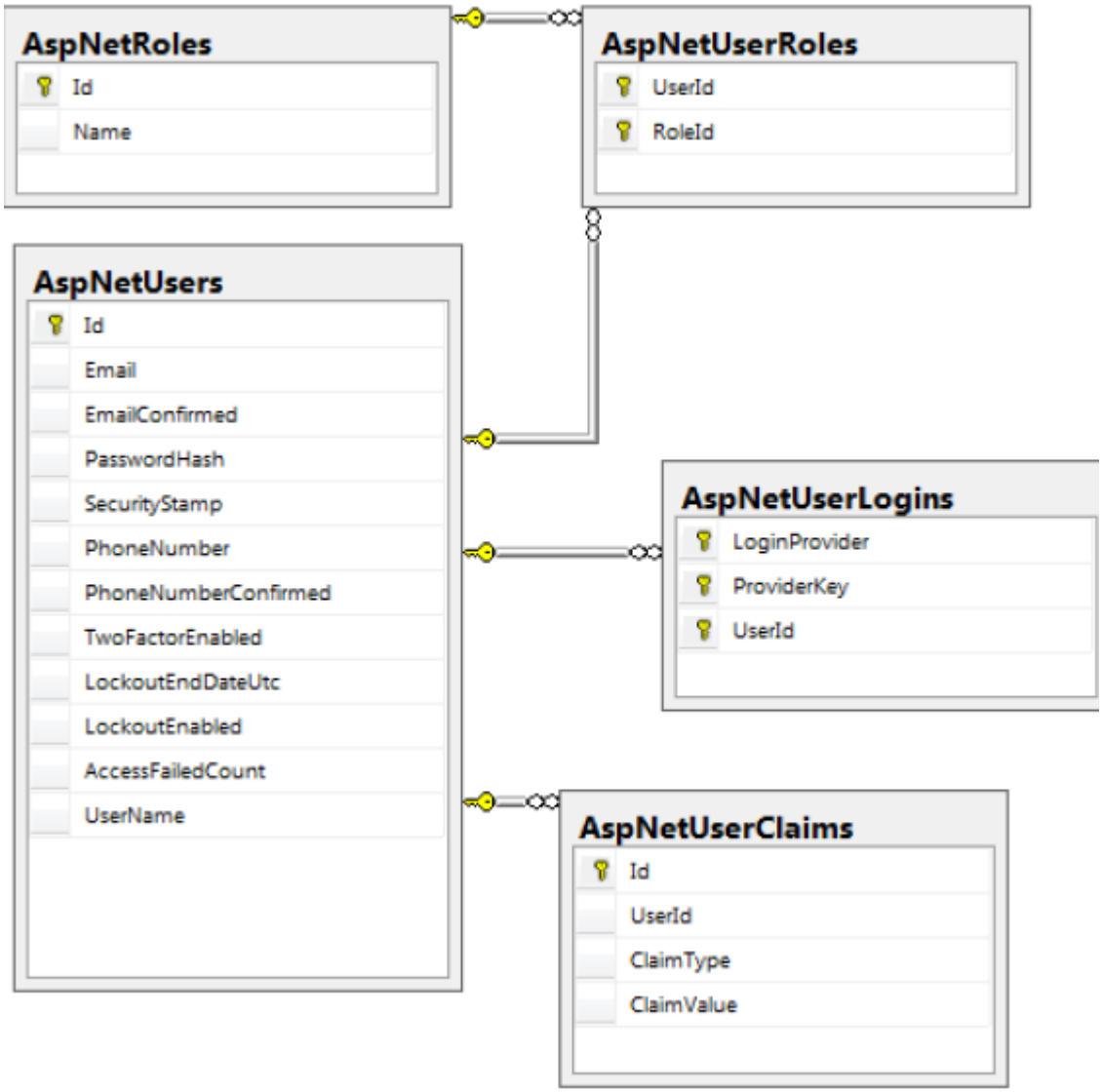


Рисунок 19. Схема базы данных приложения Bookstore. Часть 2

2.5. Структура Фреймворка

Исходя из выбора архитектуры MVVM для построения данного Фреймворка, изначально для решения было создано три проекта: модель, модель представления и представление. В качестве модели и модели представления послужили стандартные библиотеки классов C#. Также для разработки был выбран подход ViewModel First, с помощью которого можно реализовать удобную поддержку дочерних окон. Для приложения Bookstore разработано три различные модели представления. При этом какой-либо разработчик, который будет пользоваться Фреймворком, может проектировать любое количество своих моделей представления. Созданные ViewModels были унаследованы от класса

ChildViewModelBase, который поставляется в свободно распространяемом Фреймворке IkitMita.Mvvm.ViewModels, добавленным в проект с помощью менеджера загрузок. Также модели представления были помечены атрибутом Export, с помощью которого они добавляются в IoC-контейнер и впоследствии соединяются со своими представлениями. Диаграмма классов моделей представления для приложения Bookstore представлена на рисунке 20.

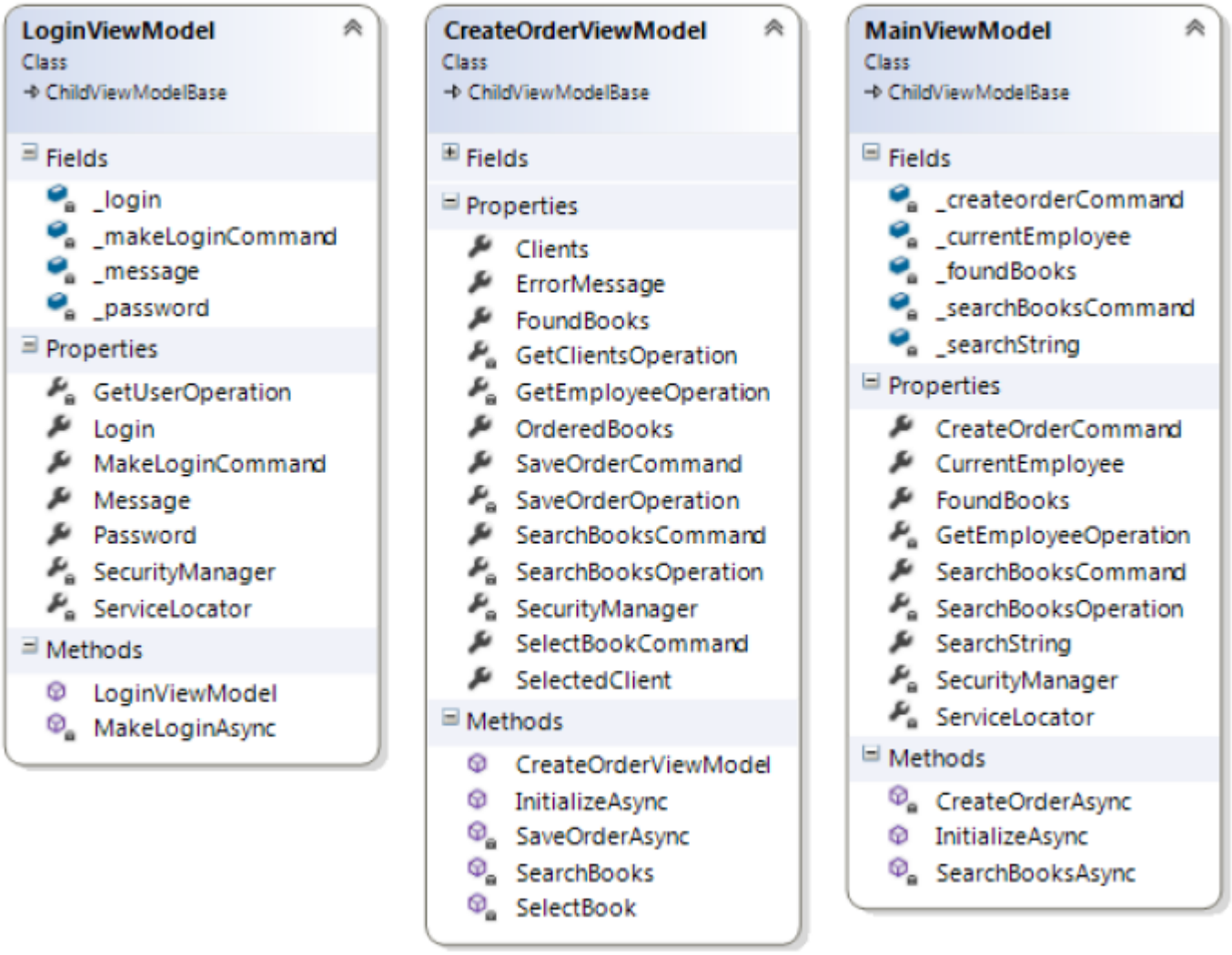


Рисунок 20. Диаграмма классов моделей представления для приложения Bookstore

Представления были разработаны в виде WPF проектов. Они проектировались на языке разметки XAML. Представлений создано ровно такое же количество, как и моделей представления. Они также были помечены атрибутом Export для связи со своими моделями представления в IoC-контейнере под контрактом интерфейса IView, который является частью свободно распространяемого Фреймворка IkitMita.Mvvm.Views. На рисунке 21 представлено View для авторизации в

приложении Bookstore.

Image not found or type unknown

Рисунок 21. Авторизация приложения Bookstore

На рисунке 22, изображено представление, которое является главной формой приложения Bookstore.

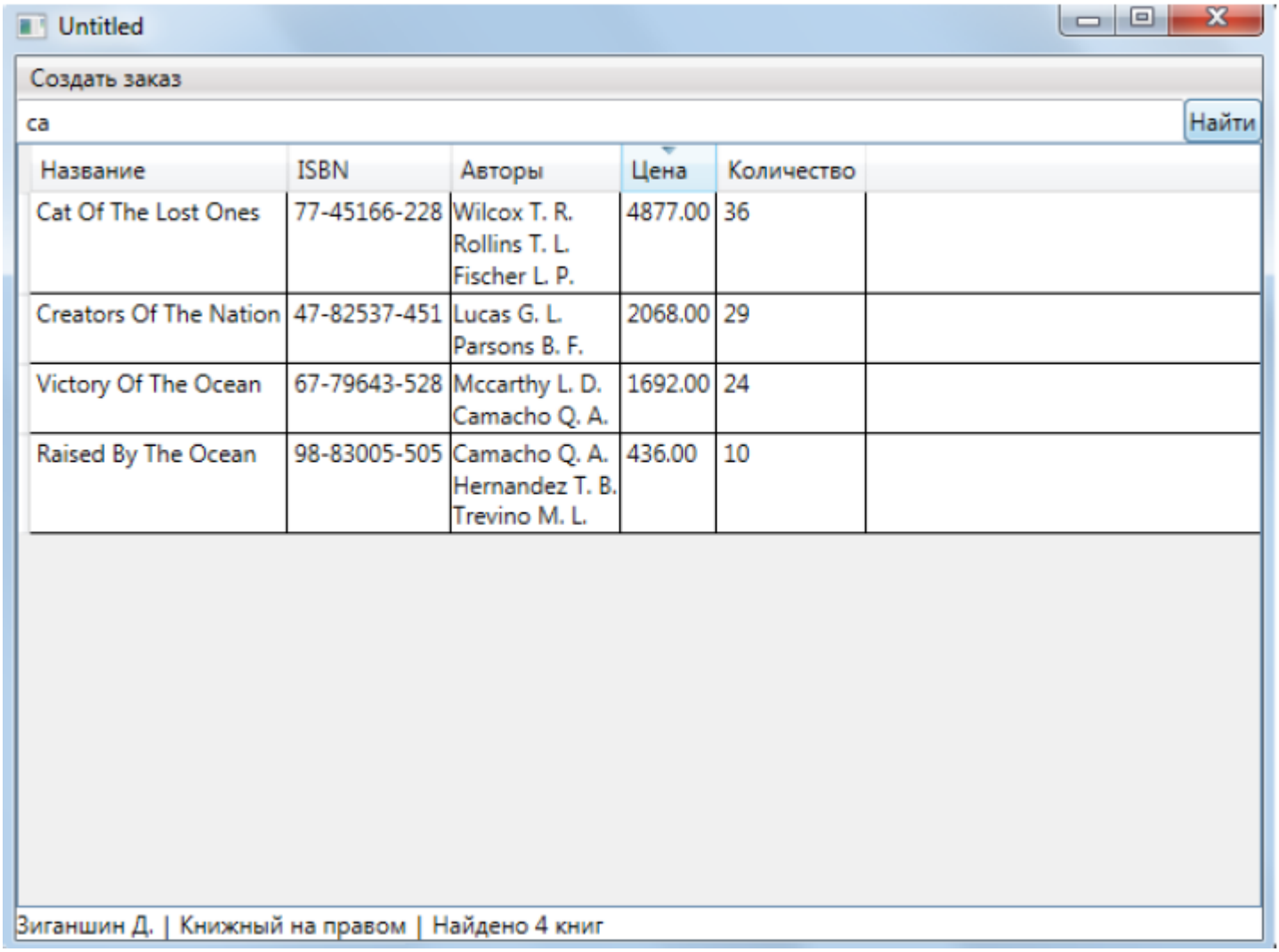


Рисунок 22. Основная форма приложения Bookstore

На рисунке 23 представлено View для создания заказа в приложении Bookstore.

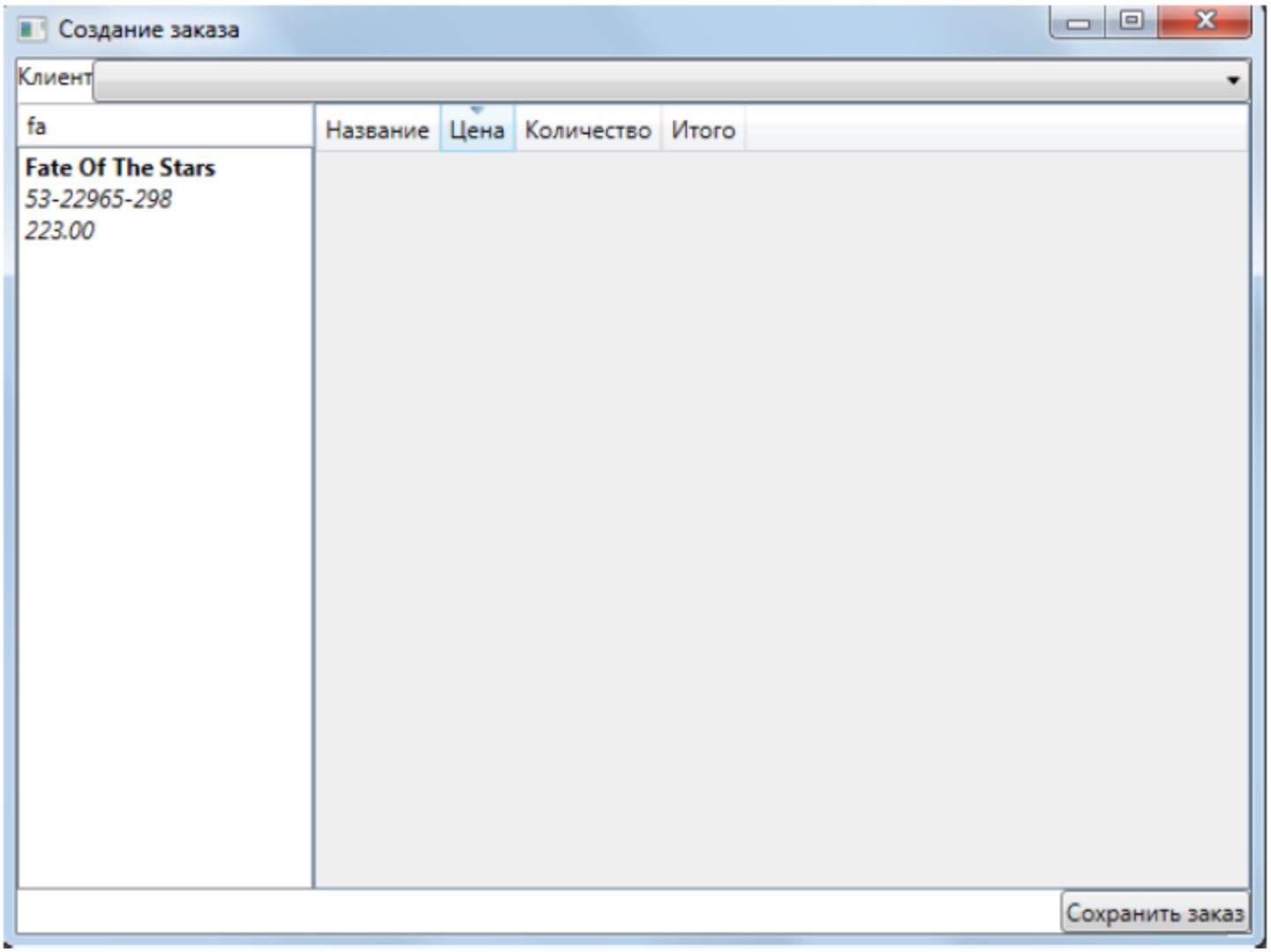


Рисунок 23. Создание заказа в приложении Bookstore

Во фреймворке был разработан специальный проект для защиты данных. В нем представлен интерфейс ISecurityManager с методами, которые обеспечивают корректную авторизацию пользователя и проверку его роли в

системе. Также здесь представлен класс PasswordManager, методы которого

хэшируют пароли. С помощью данной технологии процесс взлома становится трудоемкой задачей. Какой-либо злоумышленник не сможет открыть таблицы в базе данных и посмотреть пароли, так как они хранятся в хешированном виде. На рисунке 24 представлена диаграмма классов данного проекта.

Image not found or type unknown

Рисунок 24. Диаграмма классов для защиты информации

Исходя из того, что Фреймворк будет предназначен для построения многозвенной распределенной информационной системы, одной из ключевых его частей является проект веб-сервиса. Пользовательские клиентские приложения, построенные на описываемом Фреймворке будут взаимодействовать не с конкретной СУБД, а именно с веб - сервисом. То есть данные, полученные из БД, будут сначала обрабатываться на дополнительном сервере и только потом поступать на клиентскую сторону. Проект веб - сервиса был разработан на шаблоне MVC. Контроллеры в данном случае исполняли роль посредников при передаче моделей из клиентской стороны на сторону базы данных и обратно с их обработкой на промежуточном сервере. На рис. 25 представлен фрагмент промежуточного веб - сервера, на котором происходит обработка моделей.

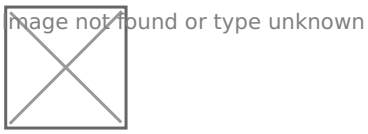


Рисунок 25. Фрагмент веб – сервиса

Также ключевой частью Фреймворка является проект DataAccess.WebApi, который обеспечивает доступ к данным на уже упомянутом веб - сервисе. Структура данного проекта представлена на рисунке 26.

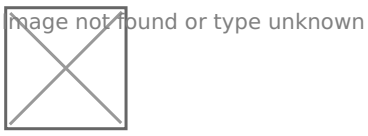


Рисунок 26. Структура проекта DataAccess.WebApi

Данный проект содержит класс AuthModel и интерфейс IAuthTokenProvider, которые предоставляют токен для обеспечения безопасности пользователя при авторизации и его идентификации. Также здесь присутствует класс SecurityManager, реализующий вышеупомянутый интерфейс ISecurityManager для обеспечения корректной авторизации пользователей. На рисунке 27 изображена диаграмма классов проекта DataAccess.WebApi.

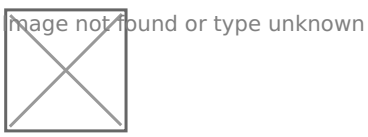


Рисунок 27. Диаграмма классов проекта DataAccess.WebApi

Ключевую роль здесь играет абстрактный класс WebApiClient. В нем представлены основные методы, которые отвечают за обработку данных на промежуточном веб - сервере. От него наследуются классы с методами, реализующими операции для приложения Bookstore. Любой разработчик, классы для обеспечения каких-либо операций, унаследовать эти классы от WebApiClient, и данные операции будут обрабатываться уже на веб-сервере.

Таким образом, был разработан свободно распространяемый Фреймворк для проектирования распределенной информационной системы. На первом этапе были рассмотрены основные инструментальные средства и

выбраны оптимальные. На дальнейшем происходила разработка приложения Bookstore, которое послужило отправной точкой для проектирования Фреймворка. В дальнейшем, можно будет создавать другие приложения, для которых каркасом будет служить разработанный Фреймворк.

ЗАКЛЮЧЕНИЕ

Пользовательский интерфейс, интерфейс пользователя – одна из разновидностей интерфейсов, который является совокупностью средств и методов взаимодействия пользователя с вычислительными устройствами (в частности, ПК).

Анализ первой главы курсовой работы на тему «Варианты построения интерфейса программ: особенности и эволюция» выявил следующее:

Во- первых, Интерфейсы выполняют конкретные задачи и их эффективность измерима. Однако они могут выходить за рамки чисто практического применения. Лучшие интерфейсы те, которые вдохновляют, пробуждают чувства, удивляют и усиливают наш опыт общения с миром.

Во-вторых, за более чем полвека своего существования ЭВМ проделали огромный путь, развившись в сложные и мощные системы, нашедшие применение практически во всех современных отраслях. Если присмотреться к интерфейсам современных операционных систем, то можно заметить явные сходства с графическими оболочками родоначальников всех GUI Xerox Alto и Apple Lisa. Да, первые графические интерфейсы были примитивны, но была в них одновременно и та подкупающая простота, которой так иногда не хватает перегруженным

визуальными эффектами оболочкам современных программ и операционных систем.

Во второй главе курсовой работы был спроектирован свободно распространяемый Фреймворк для построения распределенной информационной системы. Ключевой особенностью разработанного продукта является то, что в будущем он будет выступать каркасом для проектирования приложений с распределенной обработкой данных. Благодаря этому, большое количество разработчиков, сэкономит немало времени для решения задач, связанных с их собственной бизнес - логикой.

Результатом курсовой работы стал программный продукт – Фреймворк для построения распределенной ИС. Разработка велась на платформе .NET Framework на языке программирования C# с использованием технологии WPF. Для осуществления доступа к данным были использованы Entity Framework и WebApi.

В дальнейшем развитие Фреймворка планируется его внедрение на сервис NuGet, который представляет собой систему управления пакетами. С помощью этого, программисты смогут встраивать разработанный Фреймворк в свои приложения и полностью пользоваться его функционалом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абросимова, М.А. Информационные технологии в государственном и муниципальном управлении: Учебное пособие / М.А. Абросимова. — М.: КноРус, 2015. — 248 с.
2. Агальцов, В.П. Информатика для экономистов: Учебник / В.П. Агальцов, В.М. Титов. — М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2015. — 448 с.
3. Акперов, И.Г. Информационные технологии в менеджменте: Учебник / И.Г. Акперов, А.В. Сметанин, И.А. Коноплева. — М.: НИЦ ИНФРА-М, 2015. — 400 с.
4. Антопольский, А.Б. Информационные ресурсы России: Научно-методическое пособие / А.Б. Антопольский. — М.: Либерия, 2016. — 424с.
5. Балдин, К.В. Информационные системы в экономике: Учебник / К.В. Балдин, В.Б. Уткин. — М.: Дашков и К, 2013. — 395 с.
6. Балдин, К.В. Информационные технологии в менеджменте: Учеб. для студ. учреждений высш. проф. образования / К.В. Балдин. — М.: ИЦ Академия, 2012. — 288 с.

7. Божко, В.П. Информатика: данные, технология, маркетинг / В.П. Божко, В.В. Брага, Н.Г. Бубнова. — М.: Финансы и статистика, 2017. — 224 с.
8. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. — М.: НИЦ ИНФРА-М, 2016. — 283 с.
9. Васильков, А.В. Информационные системы и их безопасность: Учебное пособие / А.В. Васильков, А.А. Васильков, И.А. Васильков. — М.: Форум, 2015. — 528 с.
10. Велихов, А. С. Основы информатики и компьютерной техники: учебное пособие / А. С. Велихов. – Москва: СОЛОН-Пресс, 2017. – 539 с.
11. Волкова В.Н. Теория информационных процессов и систем. — М.: Юрайт, 2016. — 504 с.
12. Гаврилов, М.В. Информатика и информационные технологии: Учебник для бакалавров / М.В. Гаврилов, В.А. Климов; Рецензент Л.В. Кальянов, Н.М. Рыскин. — М.: Юрайт, 2016. — 378 с.
13. Гейн, А.Г. Основы информатики и вычислительной техники / А.Г. Гейн, В.Г. Житомирский, Е.В. Линецкий, и др.. — М.: Просвещение, 2017. — 254 с.
14. Горнец, Н. Н. ЭВМ и периферийные устройства. Компьютеры и вычислительные системы: учебник для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника» / Н. Н. Горнец, А. Г. Рощин. — М.: Академия, 2017. — 240 с.: ил. — (Высшее профессиональное образование Информатика и вычислительная техника).
15. Грошев А.С. Информатика: лабораторный практикум. Сев. (Арктич.) федер. ун-т им. М.В. Ломоносова. – Архангельск: ИД САФУ, 2017. – 154 с.
16. Завгородний, В.И. Информатика для экономистов: Учебник для бакалавров / В.П. Поляков, Н.Н. Голубева, В.И. Завгородний; Под ред. В.П. Полякова. — М.: Юрайт, 2018. — 524 с.
17. Ивасенко, А.Г. Информационные технологии в экономике и управлении: Учебное пособие / А.Г. Ивасенко, А.Ю. Гридасов, В.А. Павленко. — М.: КноРус, 2017. — 158 с.
18. Информатика / Под ред. Н.В. Макаровой. — СПб.: Питер, 2018. — 160 с.
19. Информатика Учебник для вузов Макарова Н. В., Волков В. Б. Издательство: Питер, 2017 г., 575 с.
20. Информатика. Базовый курс / Под ред. С.В. Симоновича. — СПб.: Питер, 2018. — 640 с.
21. Информатика. Базовый курс: учебное пособие для высших технических учебных заведений / [С. В. Симонович и др.]. – Санкт-Петербург: Питер, 2018. – 639 с.

22. Информатика: Энциклопедический словарь для начинающих / ред. Д.А. Поспелов. — М.: Педагогика-Пресс, 2016. — 352 с.
23. Иопа, Н. И. Информатика: (для технических специальностей): учебное пособие / Н. И. Иопа. – Москва: КноРус, 2018. – 469 с.
24. Исаев, Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. — М.: Омега-Л, 2018. — 464 с.
25. Керниган, Б.У. Язык программирования С / Б.У. Керниган, Д.М. Ритчи; Пер. с англ. В.Л. Бродовой. — М.: Вильямс, 2015. — 304 с.
26. Киселев, Г.М. Информационные технологии в экономике и управлении (эффективная работа в MS Office 2007): Учебное пособие / Г.М. Киселев, Р.В. Бочкова, В.И. Сафонов. — М.: Дашков и К, 2017. — 272 с.
27. Кудряшов Б.Д. Теория информации. — СПб.: Питер, 2009. — 320 с.
28. Логинов, В.Н. Информационные технологии управления: Учебное пособие / В.Н. Логинов. — М.: КноРус, 2016. — 240 с.
29. Ляхович В.Ф., Молодцов В.А., Рыжикова Н.Б. Основы информатики. — М.: КноРус, 2017. — 348 с.
30. Макарова Н. В. Информатика и информационно-коммуникационные технологии.- СПб.: Питер, 2017. — 224 с.
31. Михеева Е.В. Информационные технологии в профессиональной деятельности. — М.: Академия, 2018. — 384 с.
32. Патрушина, С.М. Информационные системы в экономике: Учебное пособие / С.М. Патрушина, Н.А. Аручиди. — М.: Мини Тайп, 2016. — 144с.
33. Просветов Г.И. Анализ данных с помощью Excel. Задачи и решения. — М.: Альфа-Пресс, 2015. — 160 с.
34. Светлов, Н.М. Информационные технологии управления проектами: Учебное пособие / Н.М. Светлов, Г.Н. Светлова. — М.: НИЦ ИНФРА-М, 2017. — 232 с.
35. Страуструп, Б. Язык программирования С++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. — М.: БИНОМ, 2016. — 1136 с.
36. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен; Пер. с англ. Ю.Н. Артеменко. — М.: Вильямс, 2016. — 1312 с.
37. Угринович, Н. Информатика и информационные технологии / Н. Угринович. — М.: Бином. Лаборатория знаний, 2017. — 512 с.
38. Федотова Е. Л. Информатика: курс лекций / Е. Л. Федотова, А. А.
39. Федотова, Е.Л. Информационные технологии в науке и образовании: Учебное пособие / Е.Л. Федотова, А.А. Федотов. — М.: ИД ФОРУМ, НИЦ ИНФРА-М, 2016. — 336 с.

40. Хейлсберг, А. Язык программирования C#. Классика Computers Science / А. Хейлсберг, М. Торгерсен, С. Вилтамут. — СПб.: Питер, 2018. — 784 с.
41. Щипицина, Л.Ю. Информационные технологии в лингвистике: Учебное пособие / Л.Ю. Щипицина. — М.: Флинта, Наука, 2016. — 128 с.
42. Ясенев, В.Н. Информационные системы и технологии в экономике.: Учебное пособие для студентов вузов / В.Н. Ясенев. — М.: ЮНИТИ-ДАНА, 2016. — 560 с.

1. Горнец, Н. Н. ЭВМ и периферийные устройства. Компьютеры и вычислительные системы: учебник для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника» / Н. Н. Горнец, А. Г. Рощин. — М.: Академия, 2017. — 240 с.: ил. — (Высшее профессиональное образование Информатика и вычислительная техника). ↑

2. Керниган, Б.У. Язык программирования C / Б.У. Керниган, Д.М. Ритчи; Пер. с англ. В.Л. Бродовой. — М.: Вильямс, 2015. — 304 с. ↑

3. Киселев, Г.М. Информационные технологии в экономике и управлении (эффективная работа в MS Office 2007): Учебное пособие / Г.М. Киселев, Р.В. Бочкова, В.И. Сафонов. — М.: Дашков и К, 2017. — 272 с. ↑

4. Балдин, К.В. Информационные технологии в менеджменте: Учеб. для студ. учреждений высш. проф. образования / К.В. Балдин. — М.: ИЦ Академия, 2017. — 288 с. ↑

5. Велихов, А. С. Основы информатики и компьютерной техники: учебное пособие / А. С. Велихов. – Москва: СОЛОН-Пресс, 2017. – 539 с. ↑

6. Информатика / Под ред. Н.В. Макаровой. — СПб.: Питер, 2018. — 160 с. ↑

7. Исаев, Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. — М.: Омега-Л, 2018. — 464 с. ↑

8. Логинов, В.Н. Информационные технологии управления: Учебное пособие / В.Н. Логинов. — М.: КноРус, 2016. — 240 с. ↑

9. Патрушина, С.М. Информационные системы в экономике: Учебное пособие / С.М. Патрушина, Н.А. Аручиди. — М.: Мини Тайп, 2016. — 144с. ↑

10. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. ↑

11. Угринович, Н. Информатика и информационные технологии / Н. Угринович. — М.: Бином. Лаборатория знаний, 2017. — 512 с. ↑

12. Хейлсберг, А. Язык программирования C#. Классика Computers Science / А. Хейлсберг, М. Торгерсен, С. Вилтамут. — СПб.: Питер, 2018. — 784 с. ↑

13. Светлов, Н.М. Информационные технологии управления проектами: Учебное пособие / Н.М. Светлов, Г.Н. Светлова. — М.: НИЦ ИНФРА-М, 2017. — 232 с. ↑

14. Ясенев, В.Н. Информационные системы и технологии в экономике.: Учебное пособие для студентов вузов / В.Н. Ясенев. — М.: ЮНИТИ-ДАНА, 2016. — 560 с. ↑

15. Макарова Н. В. Информатика и информационно-коммуникационные технологии.- СПб.: Питер, 2017. — 224 с. ↑

16. Керниган, Б.У. Язык программирования C / Б.У. Керниган, Д.М. Ритчи; Пер. с англ. В.Л. Бродовой. — М.: Вильямс, 2015. — 304 с. ↑

17. Завгородний, В.И. Информатика для экономистов: Учебник для бакалавров / В.П. Поляков, Н.Н. Голубева, В.И. Завгородний; Под ред. В.П. Полякова. — М.: Юрайт, 2018. — 524 с. ↑

18. \? Информатика / Под ред. Н.В. Макаровой. — СПб.: Питер, 2018. — 160 с. ↑

19. Гейн, А.Г. Основы информатики и вычислительной техники / А.Г. Гейн, В.Г. Житомирский, Е.В. Линецкий, и др.. — М.: Просвещение, 2017. — 254 с. ↑

20. Грошев А.С. Информатика: лабораторный практикум. Сев. (Арктич.) федер. ун-т им. М.В. Ломоносова. – Архангельск: ИД САФУ, 2017. – 154 с. ↑

21. Исаев, Г.Н. Информационные технологии: Учебное пособие / Г.Н. Исаев. — М.: Омега-Л, 2018. — 464 с. ↑

22. Светлов, Н.М. Информационные технологии управления проектами: Учебное пособие / Н.М. Светлов, Г.Н. Светлова. — М.: НИЦ ИНФРА-М, 2017. — 232 с. ↑

23. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. ↑

24. Федотова Е. Л. Информатика: курс лекций / Е. Л. Федотова, А. А. ↑

25. Хейлсберг, А. Язык программирования C#. Классика Computers Science / А. Хейлсберг, М. Торгерсен, С. Вилтамут. — СПб.: Питер, 2018. — 784 с. ↑

26. Ясенев, В.Н. Информационные системы и технологии в экономике.: Учебное пособие для студентов вузов / В.Н. Ясенев. — М.: ЮНИТИ-ДАНА, 2016. — 560 с. ↑

27. Информатика Учебник для вузов Макарова Н. В., Волков В. Б. Издательство: Питер, 2017 г., 575 с. ↑

28. Завгородний, В.И. Информатика для экономистов: Учебник для бакалавров / В.П. Поляков, Н.Н. Голубева, В.И. Завгородний; Под ред. В.П. Полякова. — М.: Юрайт, 2018. — 524 с. ↑

29. Варфоломеева, А.О. Информационные системы предприятия: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. — М.: НИЦ ИНФРА-М, 2016. — 283 с. ↑

30. Антопольский, А.Б. Информационные ресурсы России: Научно-методическое пособие / А.Б. Антопольский. — М.: Либерия, 2016. — 424с. ↑

31. Божко, В.П. Информатика: данные, технология, маркетинг / В.П. Божко, В.В. Брага, Н.Г. Бубнова. — М.: Финансы и статистика, 2017. — 224 с. ↑

32. Волкова В.Н. Теория информационных процессов и систем. — М.: Юрайт, 2016. — 504 с. ↑

33. Горнец, Н. Н. ЭВМ и периферийные устройства. Компьютеры и вычислительные системы: учебник для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника» / Н. Н. Горнец, А. Г. Рощин. — М.: Академия, 2017. — 240 с.: ил. — (Высшее профессиональное образование Информатика и вычислительная техника). ↑

34. Информатика / Под ред. Н.В. Макаровой. — СПб.: Питер, 2018. — 160 с. ↑

35. Логинов, В.Н. Информационные технологии управления: Учебное пособие / В.Н. Логинов. — М.: КноРус, 2016. — 240 с. ↑

36. Патрушина, С.М. Информационные системы в экономике: Учебное пособие / С.М. Патрушина, Н.А. Аручиди. — М.: Мини Тайп, 2016. — 144с. ↑

37. Страуструп, Б. Язык программирования C++: Специальное издание / Б. Страуструп; Пер. с англ. Н.Н. Мартынов. — М.: БИНОМ, 2016. — 1136 с. ↑