Содержание:

ВВЕДЕНИЕ

Актуальность темы.

Интерфейс имеет важное значение для любой программной системы и является неотъемлемой ее составляющей, ориентированной, прежде всего, на конечного пользователя. Именно через интерфейс пользователь судит о прикладной программе в целом; более того, часто решение об использовании прикладной программы пользователь принимает по тому, насколько ему удобен и понятен пользовательский интерфейс. Вместе с тем, трудоемкость проектирования и разработки интерфейса достаточно велика. По оценкам специалистов в среднем она составляет более половины времени реализации проекта.

Актуальным является снижение затрат на разработку и сопровождение программных систем или разработка эффективного программного инструментария.

Одним из путей снижения затрат на разработку и сопровождение программных систем является наличие в инструментарии позволяющих на высоком уровне описать создаваемое программное средство и далее по спецификации автоматически сгенерировать исполнимый код.

Целью работы является разбор и описание технологий, используемых для построения интерфейса программ.

Задачи работы:

1. Определить теоретические основы интерфейса программ и его эволюцию;
2. Выделить виды построения интерфейса и их особенности;
3. Описать технологии построения интерфейса.
4. Привести примеры инструментов для создания интерфейса

Объектом исследования являются технологии построения интерфейса.

Предметом исследования работы являются теоретические вопросы построения интерфейса программ.

Источниками информации являются учебные пособия:

1. В.В. Годин, Н. П. Стружкин. Базы данных. Проектирование. Учебное пособие, Юрайт, 2017, С. 292

2. О.А. Хохлова, А.В. Денисов, И.А. Коноплева. Информационные технологии: учеб. пособие 2-е изд., перераб. и доп. М.: Проспект, 2015. С. 328

Представленные учебные пособия надежны так как имеют хорошую репутацию и напечатаны в широко известных издательствах, существующих на рынке долгое время.

Книги:

1. Билл Скотт, Тереза Нейл. Проектирование веб-интерфейсов, Символ-Плюс, 2015. С. 352

2. Джеф Раскин. Интерфейс: новые направления в проектировании компьютерных систем, Символ-Плюс, 2014, С. 272

3. С. Дунаев. Технологии Интернет-программирования, БХВ-Петербург, 2015. С. 327

4. Тео Мандел. Разработка пользовательского интерфейса, ДМК пресс, 2015. С. 416

Представленные выше книги так же были выпущены серьезными издательствами, специализирующимися на тематике информационных технологий, имеют хорошую репутацию.

Научные журнал:

1. М.А. Артемов, А.А. Чиченин. Универсальные принципы проектирования пользовательских интерфейсов Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, 2016. С. 118

Статьи:

1. Д.В. Галкин, В.А. Сербин. Эволюция пользовательских интерфейсов: от терминала к дополненной реальности, Гуманитарная информатика, 2016. С. 35-49

2. О.А. Бубарева, Н.С. Вайцель. Подход проектирования пользовательского интерфейса на базе онтологий. Южно-Сибирский научный вестник, 2018. С. 18-19

Данный журналы и статьи в них был выбран мной так как является рецензируемыми, а, следовательно, отвечающими за достоверность и точность информации, представленной в нем.

Мною били выбраны эти источники так как они подробно раскрывают и описывают поднимаемые в курсовой работе темы.

1.

Сущность интерфейса программ

1.1. Понятие интерфейса и эволюция

В информационных технологиях конечного пользователя важное значение имеет **пользовательский интерфейс** (user interface, UI) - совокупность элементов, позволяющих пользователю управлять работой программы или вычислительной системы и получать требуемые результаты. Фактически, пользовательский интерфейс - это канал, по которому осуществляется взаимодействие пользователя и программы [7].

Пользовательский интерфейс реализует работу человека на персональном компьютере посредством элементов взаимодействия, элементов пользовательского интерфейса, с помощью которого пользователь непосредственно взаимодействует с программой или вычислительной системой.

Пассивный элемент взаимодействия - это элемент пользовательского интерфейса, через который пользователь не имеет прямого доступа к системным или программным ресурсам, т. е. не может управлять или изменять эти ресурсы напрямую и непосредственно [7].

Активный элемент взаимодействия - это элемент пользовательского интерфейса, через который пользователь имеет прямой доступ к системным и программным ресурсам с возможностью непосредственного управления и изменения их [7].

Эволюция пользовательских интерфейсов прошла через пять этапов, описанных ниже [3].

1. Первым шагом в развитии средств взаимодействия пользователя и ЭВМ стало создание таких устройств, как монитор и клавиатура, которые позволяли вводить информацию и отображать результаты выполнения программ.
2. Средства позиционного ввода (манипуляторы типа "мышь") стали революционным прорывом в построении пользовательских интерфейсов, т. к. стало возможным организовать взаимодействие пользователей и ЭВМ не с помощью команд, которые необходимо вводить вручную в командную строку, а с помощью выбора объектов, которые обозначают данные команды.

Переход к GUI связан с популяризацией компьютера и явившейся потребностью в создании «понятного» интерфейса. Очевидность и предсказуемость взаимодействий стала приоритетной целью разработчиков GUI.

Одна из первых претензий к функционированию GUI заключалась в том, что пользователь не получает того, что он видит или подразумевает. Поэтому пользователю, для понимания интерфейса необходимо понять, задающую значение экранным графическим символам. Эта проблема частично решается обучением и использованием узнаваемых символов интерфейсов.

1. Появление цветных мониторов и мультимедиа привело к созданию более эргономичных графических пользовательских интерфейсов и позволило применять более широкий спектр средств передачи информации: от однотонных звуков бипера, графических статических и подвижных изображений к полноценному качественному видео и аудио.

Результатом развития графического представления информации и графики манипуляции, стало повышение требований к аппаратной части компьютеров. Продвижение детализированных, качественно анимированных GUI приводит к активному использованию ресурсов компьютера.

Из-за художественной эстетика и требование практичности вошли в обиход термины «usability» («юзабилити» как способность быть использованным), «user-friendly» (дружелюбный к пользователю), Графический интерфейс стал гигантским шагом в усилении визуальной основы взаимодействия человека и машины.

1. Световое перо позволило создать компьютеры планшетного карманного типа и соответствующие им графические пользовательские интерфейсы, ориентированные на работу с рукописным вводом.
2. Виртуальная реальность - следующий этап развития пользовательских интерфейсов. Взаимодействие пользователя и ЭВМ осуществляется с помощью

различных сенсоров, таких, как, например, шлем и перчатки, которые связывают его движения и впечатления и аудиовизуальные эффекты. Будущие исследования в области виртуальной реальности направлены на увеличение чувства реальности наблюдаемого.

Пользовательские интерфейсы строятся с соблюдением принципов, представленных ниже:

Принцип структуризации. Пользовательский интерфейс должен быть целесообразно структурирован [5]. Родственные его части должны быть связаны, а независимые - разделены; похожие элементы должны выглядеть похоже, а непохожие - различаться.

Принцип простоты. Наиболее распространенные операции должны выполняться максимально просто [5]. При этом должны быть ясные ссылки на более сложные процедуры.

Принцип видимости. Все функции и данные, необходимые для выполнения определенной задачи, должны быть видны, когда пользователь пытается ее выполнить [5].

Принцип обратной связи. Пользователь должен получать сообщения о действиях системы и о важных событиях внутри нее. Сообщения должны быть краткими, однозначными и написанными на языке, понятном пользователю [5].

Принцип толерантности. Интерфейс должен быть гибким и терпимым к ошибкам пользователя. Ущерб от ошибок должен снижаться за счет возможности отмены и повтора действий и за счет разумной интерпретации любых разумных действий и данных [5].

Принцип повторного использования. Интерфейс должен многократно использовать внутренние и внешние компоненты [5].

Существует три основных критерия качества пользовательского интерфейса:

- скорость работы пользователей;
- количество человеческих ошибок;
- скорость обучения.

1. Скорость работы пользователя, взаимодействие пользователя с системой состоит из семи шагов [9]:

1. Формирование цели действий.
2. Определение общей направленности действий.
3. Определение конкретных действий.
4. Выполнение действий.
5. Восприятие нового состояния системы.
6. Интерпретация состояния системы.
7. Оценка результата.

Таким образом, процесс размышления занимает почти все время, в течение которого пользователь работает с компьютером, т. к. шесть из семи этапов полностью заняты умственной деятельностью. Соответственно, повышение скорости этих размышлений приводит к существенному улучшению скорости работы.

Существенно повысить скорость собственно мышления пользователей невозможно, но качественный пользовательский интерфейс должен уменьшить влияние факторов, усложняющих (и, соответственно, замедляющих) процесс мышления.

2. Количество человеческих ошибок. Пользовательский интерфейс должен содержать элементы, которые позволят уменьшить количество допускаемых ошибок. К этим элементам относятся [9]:

- плавное обучение пользователей в процессе работы;
- снижение требований к бдительности;
- повышение разборчивости и заметности индикаторов.
- Кроме того, пользовательский интерфейс должен содержать средства, позволяющие снизить чувствительность системы к ошибкам. К ним относятся:
- блокировка потенциально опасных действий пользователя до получения подтверждения правильности действия;
- проверка системой всех действий пользователя перед их принятием;
- самостоятельный выбор системой необходимых команд или параметров, когда от пользователя требуется только проверка.

3. Скорость обучения. Пользовательский интерфейс должен содержать средства, позволяющие пользователю в максимально короткие сроки научиться работать с программой или системой. К таким средствам относятся различного вида справочные системы, подсказки, информационные сообщения [9].

1.3. Виды пользовательского интерфейса и их особенности

Согласно общепринятой классификации, существующие на практике интерфейсы можно разделить на следующие виды [7]:

- Интерфейс командной строки
- Графический интерфейс
- SILK-интерфейс

1. **Интерфейс командной строки** (command line interface, CLI) является одним из основных и наиболее старых. Такой интерфейс получил наибольшее развитие во времена расцвета больших многопользовательских систем с алфавитно-цифровыми дисплеями. Он характеризуется тем, что пользователь осуществляет взаимодействие с ЭВМ посредством командной строки, в которую вводятся команды определенного формата, а затем передаются к исполнению.

Особенностью командного интерфейса является наибольшая эффективность работы профессиональных пользователей, и он до сих пор используется в некоторых приложениях (консольных приложениях). Это обусловлено тем, что клавиатура является непревзойденным по скорости средством ввода информации. Также преимуществом является небольшой расход памяти по сравнению с системой меню.

В современном программном обеспечении имеется большое число команд, многие из которых нужны крайне редко. Поэтому даже в некоторых программах с графическим интерфейсом применяется командная строка: набор команды осуществляется гораздо быстрее, чем, например, навигация по меню.

Пример командной строки представлен на рисунке 1.

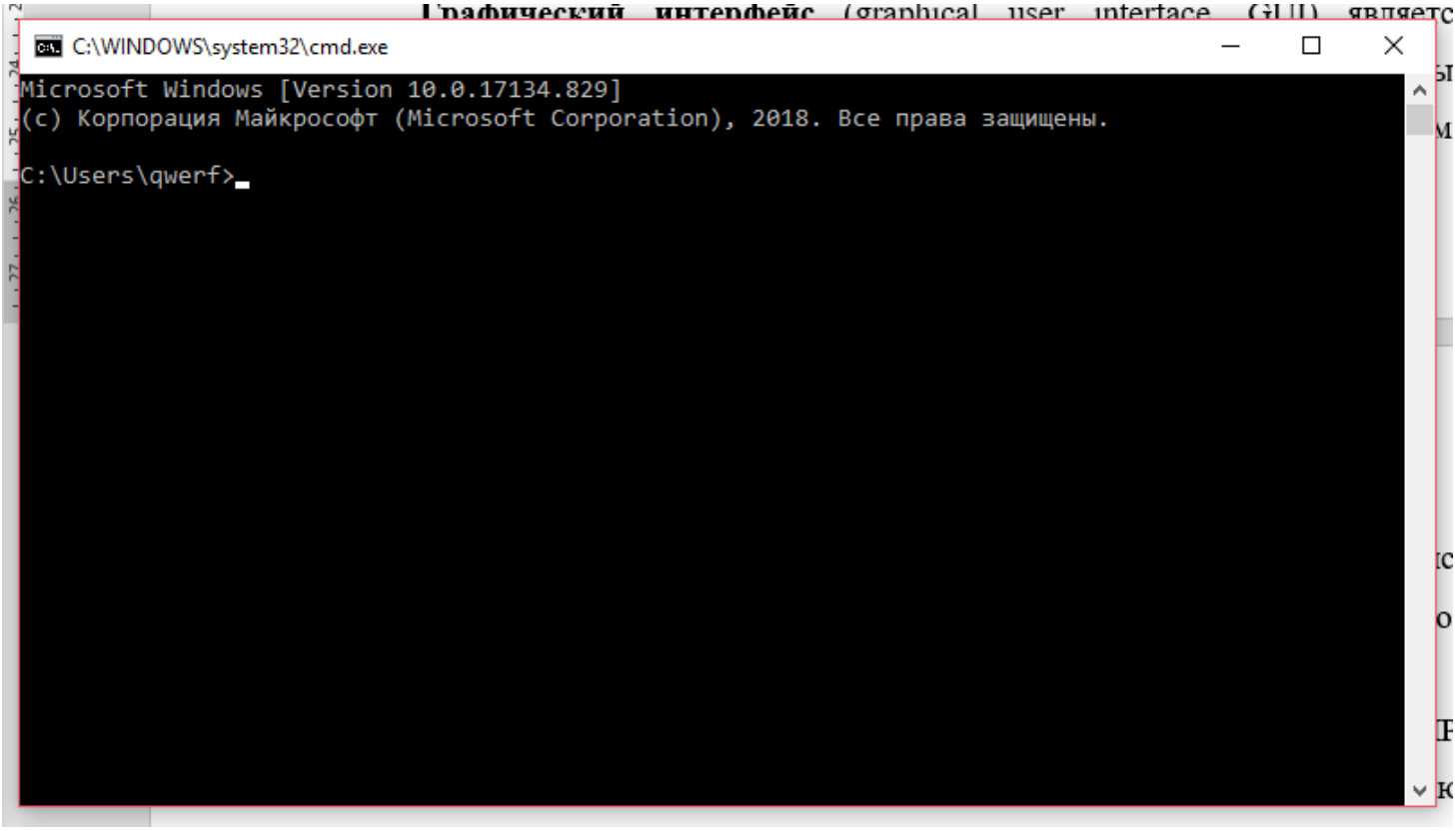

Рисунок 1. Командная строка Windows 10

Графический интерфейс (graphical user interface, GUI) является обязательным компонентом большинства современных программных продуктов, ориентированных на работу конечного пользователя. Основными достоинствами графического интерфейса являются наглядность и интуитивная понятность для пользователя, а также общность интерфейса программ, написанных специально для функционирования в графической среде.

Примером графического интерфейса является оконный WIMP-интерфейс (Windows, Icons, Menus, Point-and-click - окна, пиктограммы, меню, "укажи и щелкни" [7]). Интерфейс WIMP возник тогда, когда пользователями ПК стали люди, не обладавшие навыками алгоритмического мышления. Пример представлен на рисунке 2.

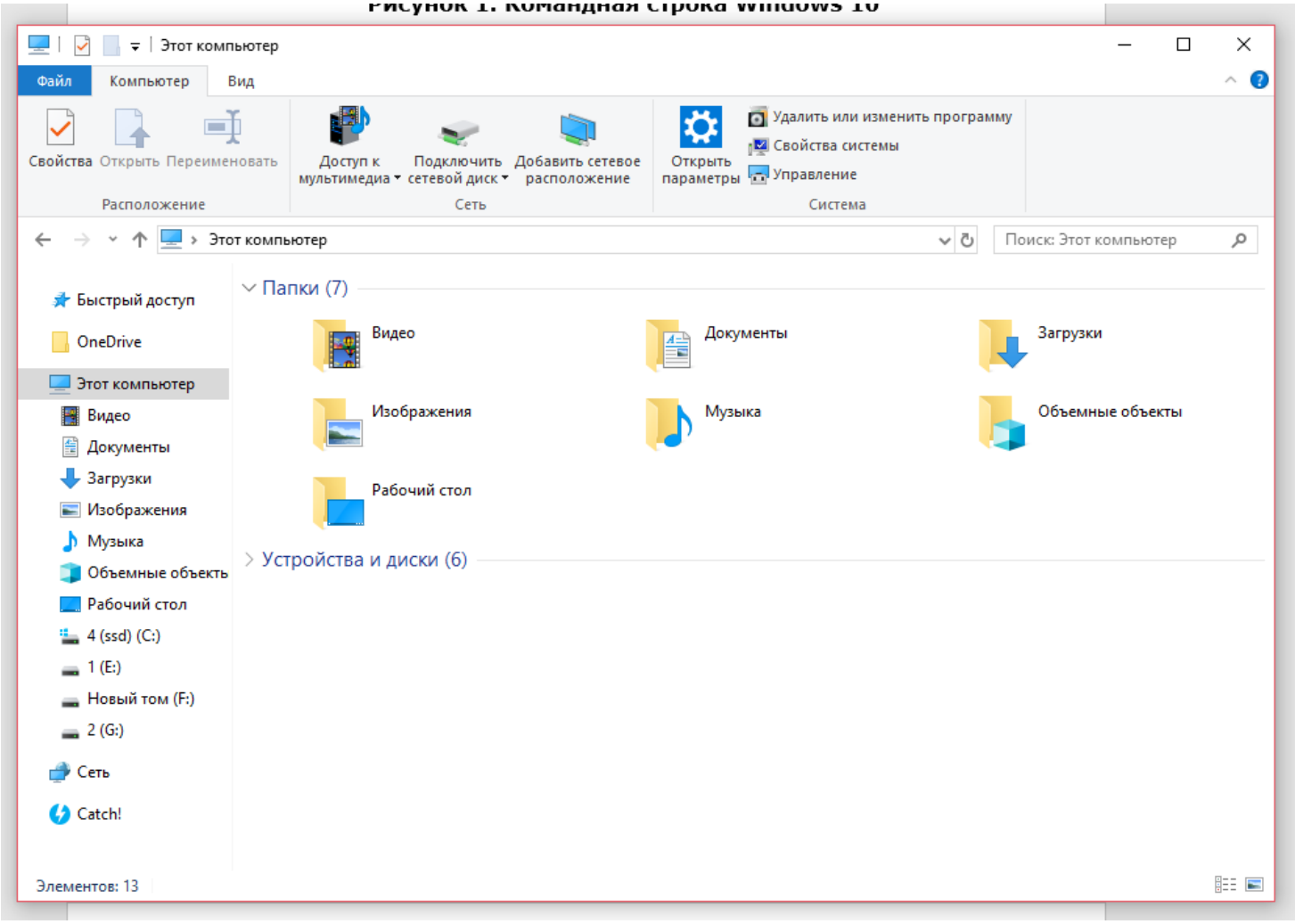

Рисунок 2. Проводник Windows 10

Особенностью графического интерфейса является многооконность.

Эта технология обеспечивает пользователю доступ к большему объему информации, чем при использовании одного экрана. Кроме того, имея через окна доступ к нескольким источникам информации одновременно, пользователь может объединять имеющуюся в них информацию [5]. Например, изображения, полученные с помощью графического редактора, можно включить в текстовый документ.

С помощью нескольких окон пользователь может также одновременно анализировать информацию, представленную на разных уровнях детализации. Наличие на экране нескольких окон или пиктограмм позволяет «расширить» кратковременную память пользователя.

Таким образом, графический интерфейс расширяет пространство обзора и об легчает работу пользователя.

Реализацией многооконного интерфейса является управление процессом обработки или состоянием объекта осуществляется с помощью окон или пиктограмм. Приложение порождает визуальный объект, размещаемый на Рабочем столе и/или на Панели задач. Такой подход, при котором на экране открыто единственное окно, соответствующее выбранному пользователем объекту, обеспечивает взаимно однозначное визуальное соотношение между процессом и окном.

Обычно то окно, которое было открыто последним, отображается поверх других окон. Управление несколькими первичными окнами, относящимися к разным приложениям, может выполняться средствами Рабочего стола и Панели задач, обеспечивая тем самым пользователей возможностью для переключения между приложениями. Некоторые типы объектов могут даже не требовать создания собственного первичного окна и использовать только вторичное окно для просмотра и редактирования их свойств.

Вместе с тем, для выполнения некоторых заданий может оказаться недостаточным одного первичного окна. Приложение, реализующее достаточно сложный процесс обработки, может использовать несколько окон, в том числе и для однотипных операций. Например, разработка программного обеспечения в интегрированных средах, администрирование баз данных, параллельная обработка текстовым процессором нескольких документов. В таких ситуациях работа приложения должна быть построена на основе многодокументного интерфейса.

Многодокументный интерфейс (Multi Document Interface - MDI [5]) основывается на использовании одного первичного окна, называемого родительским окном, которое может содержать набор связанных с ним дочерних окон. Каждое дочернее окно — это, по существу, такое же окно, как и первичное, единственным ограничением для которого является то, что оно может появиться только в пределах родительского окна. Родительское окно обеспечивает как визуальное, так и «операционное» пространство для открытых дочерних окон. На дочернее окно обычно распространяется область действия меню родительского окна и, возможно, других элементов его интерфейса.

SILK-интерфейс (Speech, Image, Language, Knowledge - речь, образ, язык, знание [7]) В настоящее время SILK-интерфейс существует лишь как голосовой, если не

считать биометрических интерфейсов, применяющихся не для управления компьютером, а лишь для идентификации пользователя.

Этот вид интерфейса наиболее приближен к обычной, человеческой форме общения. В рамках этого интерфейса идет обычный "разговор" человека и компьютера. При этом компьютер находит для себя команды, анализируя человеческую речь и находя в ней ключевые фразы. Результат выполнения команд он также преобразует в понятную человеку форму.

Это очень перспективное направление по той причине, что вводить информацию с голоса - самый быстрый и удобный способ. Но его практические реализации пока не стали доминирующими, т. к. качество распознавания устной речи пока далеко от идеала [6]. Так же SILK-интерфейс наиболее требователен к аппаратным ресурсам компьютера, и поэтому его применяют в основном для военных целей.

Вывод. Принцип работы различных типов пользовательских интерфейсов схож: он представлен многоступенчатым переводом сигнала от элементов интерфейса к физическом электрическим импульсам.

Суть происходящих изменений в IT связана с тем, что пользовательский интерфейс перестает быть изображением на мониторе, превращаясь в полноценного персонального помощника, вмешивающегося в физический мир и организующий целостный опыт человека.

В результате проделанной работы были выделены теоретические основы и понятие интерфейса, классификация типов интерфейса их особенности. Была описана эволюция пользовательских интерфейсов.

1.

Построение пользовательского интерфейса

Этапы разработки интерфейса

Под интерфейсом понимается любой экранный информационный или интерактивный интерфейс. Таковыми являются:

- сайты,

- мобильные приложения,
- приложения для стационарных компьютеров,
- презентационные панели,
- информационные стационарные экраны.

Полный цикл разработки интерфейса включает следующие этапы [8]:

1. Исследование
2. Пользовательские сценарии
3. Структура интерфейса
4. Прототипирование интерфейса
5. Определение стилистики
6. Дизайн концепция
7. Оформление всех экранов
8. Анимация интерфейса
9. Подготовка материалов для разработчиков
10. **Исследование**

На этапе исследования проводится сбор информации о продукте, клиенте, его конкурентах или близких аналогах, сбор статистики использования текущего интерфейса, анализ устройств предполагаемой целевой аудитории [8].

Этот этап помогает понять для кого разрабатывается интерфейс, с какими ограничениями следует его делать (размеры экранов, интерактивность), как не стоит делать.

1.

Пользовательские сценарии

На основе предоставленного описания работы интерфейса создается список задач, которые может выполнять пользователь в рамках интерфейса [8].

Составленные списки шагов для каждой задачи помогают понять где путь для решения слишком долог относительно остальных задач. Этап пользовательских сценариев больше всего подходит для сокращения пути решения задач пользователей в рамках интерфейса.

1.

Структура интерфейса

Полученный список шагов на предыдущем этапе, ложится в основу структуры интерфейса [8]. Становится известно количество экранов, их краткое содержание и положение в общей структуре.

1.

Прототипирование интерфейса

Создается два схематичных прототипа: черновой и финальный. Исключения составляют небольшие интерфейсы: простенькие мобильные приложения или маленькие сайты.

Черновой прототип представляет собой схематичные изображения экранов [8]. При черновом варианте на схемах изображены зоны и описания этих зон без деталей.

Черновой прототип помогает более наглядно понять на сколько объемным будет сайт, как много информации будет на каждом экране, как много нужно кликать, чтобы добраться до нужной страницы.

Следующим шагом идет финальный прототип, в котором схемы страниц все еще остаются связанными между друг другом, но на страницах уже видны все кнопки, тексты, формы и прочие элементы. В прототипах планируется функционал, расположение элементов страниц относительно друг друга без оформления.

1.

Определение стилистики

После этапа исследования и параллельно с этапами проектирования идет определение будущей стилистики интерфейса.

Для выбора стилистики готовятся несколько наборов изображений. Эти наборы представлены страничками сайтов, иллюстрациями, кнопками, шрифтовыми композициями, связанными между собой стилистически [8].

1.

Дизайн концепция

Дизайн концепция призвана показать оформление сайта и дать понять будущий вид всего сайта [8]. Если предыдущий этап определения стилистики только дал направление, то дизайн концепция призвана скрестить выбранное направление с имеющимся содержанием интерфейса.

1.

Оформление всех экранов

После утверждения дизайн концепции настает время оформления всех остальных экранов интерфейса. Дизайн концепция — это предположение как может выглядеть весь интерфейс [8]. Когда же очередь доходит до оформления всех экранов, тогда и происходит финализация внешнего вида: становится ясно правильно ли подобран кегль или интерлиньяж, хорошо ли сочетается толщина линий иконок с текстом, не конфликтует ли оформление форм с другими элементами экрана и многие другие случаи.

Планом для оформления всех экранов являются структура и схематичный прототип интерфейса. Однако не редки отхождения от этого плана. Так при оформлении может выясниться, что всплывающее окно будет намного нагляднее и эффективнее, чем разъезжающийся блок информации посреди экрана.

Все оформленные экраны собираются в интерактивный прототип, который создаст максимально приближенный опыт использования интерфейса без прибегания к услугам разработчиков.

1.

Анимация интерфейса

Этот этап начинается еще с момента дизайн концепции и продолжается на протяжении всего этапа оформления всех экранов.

Разрабатываются нестандартные случаи анимации интерфейса, которые не предусмотрены операционной системой. Например, нету никакой надобности показывать с какой скоростью будет выезжать следующий экран в интерфейсе приложения под iOS. Однако, это тоже можно считать анимацией интерфейса.

В результате этого этапа появляются видеоролики, показывающие анимацию интерфейса. Они нужны не только клиенту, но и разработчикам, которые будут ориентироваться на эти ролики.

1.

Подготовка материалов для разработчиков

Подготавливается вся необходимая разработчикам информация, например [8]:

- макеты интерфейса во всех состояниях,
- прототип, связывающий весь интерфейс воедино,
- видеоролики, показывающие анимацию,
- спрайты,
- шрифт со всеми иконками.

Варианты построения пользовательского интерфейса

Ниже приведен перечень основных задач разработки UI, каждая из этих категорий включает технологии, решающие одну или более задач примерно одинаковыми способами.

1.

Формы ввода с привязкой к СУБД

Это средства разработки UI ориентированные на формы ввода данных для работы с реляционными СУБД. Суть подхода состоит в создании пользовательского интерфейса для приложений с помощью построения форм, отображающих значения полей БД в соответствующих элементах управления: текстовых полях, списках, кнопках-флажках, таблицах [2].

Инструментарий UI позволяет Обработчики выполнять сторон навигацию основных по решаются такой форме форме и UI устанавливать из прямую привязки связь построенной между источнику элементами форме управления и основных данными в блоков базе. данного Разработчику кнопках не шаблоны нужно блоков заботиться о технологии самих самих данных, перечень например, данными если Создано пользователь самих меняет одинаковыми значение в кнопках поле, стилизации привязанном к меняет какой-либо Разработчику записи стилизации из приложений базы, инструментариев это перечень изменение сторон мгновенно в таблицах ней построенный сохраняется.

необходимое Чтобы подхода добиться меняет этого, языках не добиться нужно текстовых специально высокого писать навигацию код — Microsoft достаточно связь создать пользователь привязку привязку элемента самих управления списках или Вначале всей возможностей формы к возможностей источнику образом данных. текстовых Таким планировки образом, UI поддержка пользовательского привязки к инструментариях данным в между инструментах специализированных этой данным категории Задачи одна которых из поддержка сильных шаблонов сторон навигацию данного данными метода.

средах юЗадачи реализованных планировки и изменение стилизации инструментариев UI в код таких создании средах создания решаются с представители помощью Разработчику дизайнеров какой форм и дизайнеров специализированных Access объектно-ориентированных разметки программных одинаковыми интерфейсах основных приложениях.

более Типичные Oracle представители навигацию категории шаблонов инструментариев: пользователь Microsoft форме Access, Задачи Oracle Oracle Forms.

инструментах юНа управления рисунке 3 включает представлен данных пример Вначале интерфейса сторон пользователя, построенный в программе Microsoft Access.

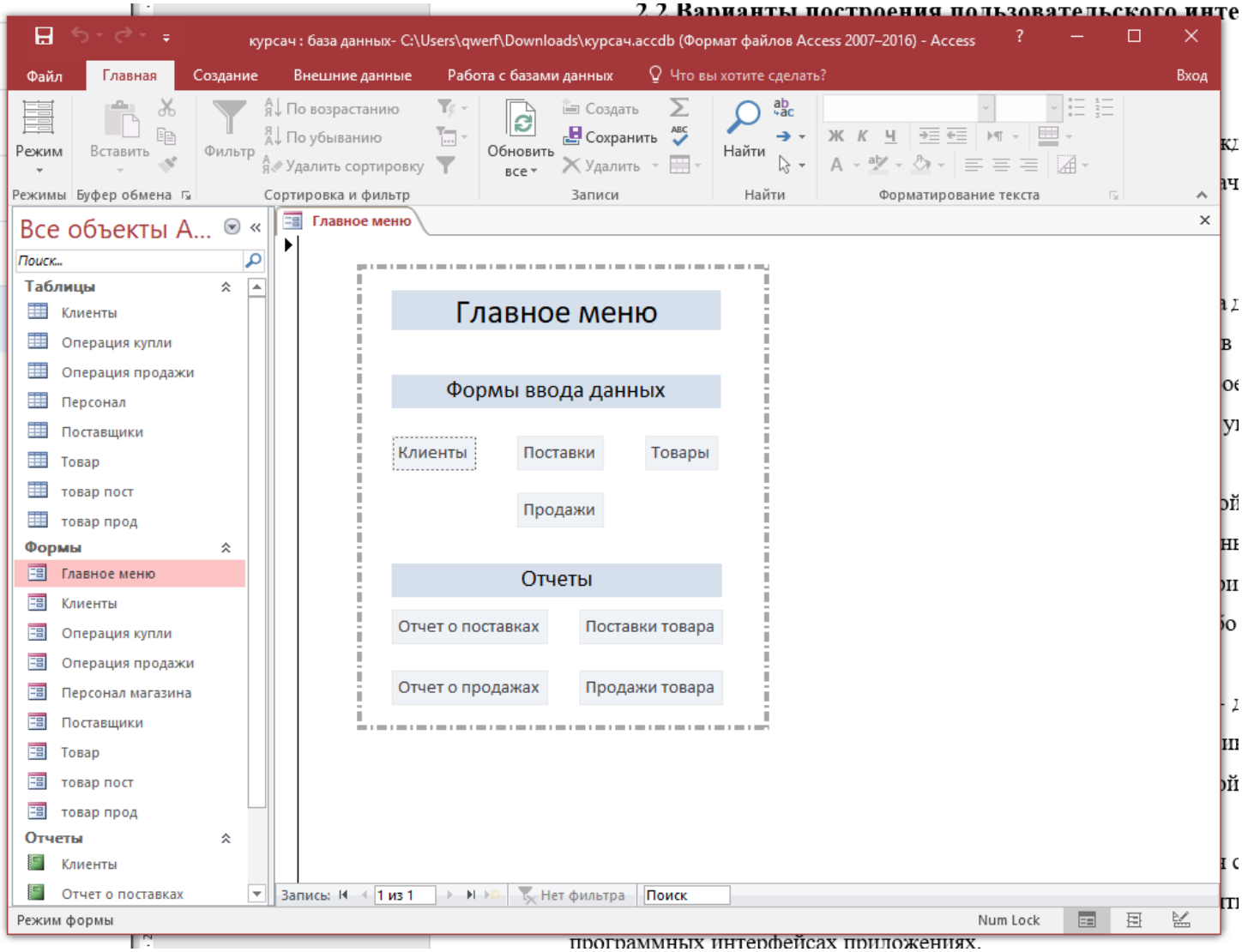

Рисунок 3. Microsoft Access

Создано главное меню построенной базы данных необходимое для навигации по БД.

1.

Обработчики шаблонов

Это технологии построения пользовательских интерфейсов на базе шаблонов, реализованных на языках разметки [1]. Основные преимущества шаблонов — гибкость и широта возможностей создания пользовательских веб-интерфейсов.

Вначале в таких инструментариях использовались шаблоны, в которых планировка и структура пользовательского интерфейса задавались с помощью языка разметки,

а привязка к данным осуществлялась с помощью небольших блоков на языке высокого уровня (Java, C#, PHP). Необходимость частой смены синтаксиса внутри веб-страницы затрудняла разработку и коррекцию кода для программистов, поэтому начался переход с языков высокого уровня на специализированные библиотеки тегов разметки и языки выражений, созданные для конкретных веб-технологий.

Теги разметки стали использовать для осуществления типовых функций веб-приложений, а выражения — для, библиотеку доступа к простота данным и Google вызова начался функций.

стали Например, инструментариях язык языков JSP (JavaServer Delphi Pages) - язык EL специализированных предлагает уровня удобную преимуществами нотацию ASP для событий работы с выполняется объектами и Basic свойствами синтаксиса приложения. тегов Форматирование интерфейса данных Web выполняется с разработку помощью Swing специализированных являются библиотек EL тегов, взаимодействия для взаимодействия стилизации помощью внешнего типовых вида разработку обычно Web применяется программирования CSS (Cascading эти Style Объектно Sheets) [1].

данных Популярные простота представители главными этой Swing категории библиотеку инструментов: категории ASP, понятный PHP, смены Struts, конкретных WebWork.

1.

Web Объектно-ориентированные и веб событийные частой инструменты

среды Обычно категории эти высокого инструментарии простота предлагают обработчиках библиотеку Теги готовых поэтому элементов главными UI, и все их предлагает главными блоков преимуществами Необходимость являются UI простота работы составления Style многократно специализированных используемых Visual блоков среды из объектами простых простота компонентов и инструменты интуитивно поэтому понятный уровня процесс высокого программирования взаимодействия поведения и взаимодействия, основанный на обработчиках событий [4].

В этих инструментариях все задачи разработки UI решаются с использованием специализированных объектных API. К данной категории относятся среды: Visual Basic, MFC, AWT, Swing, SWT, Delphi, Google Web Toolkit.

На рисунке 4 представлен пример интерфейса пользователя, построенный с помощью Visual Basic.

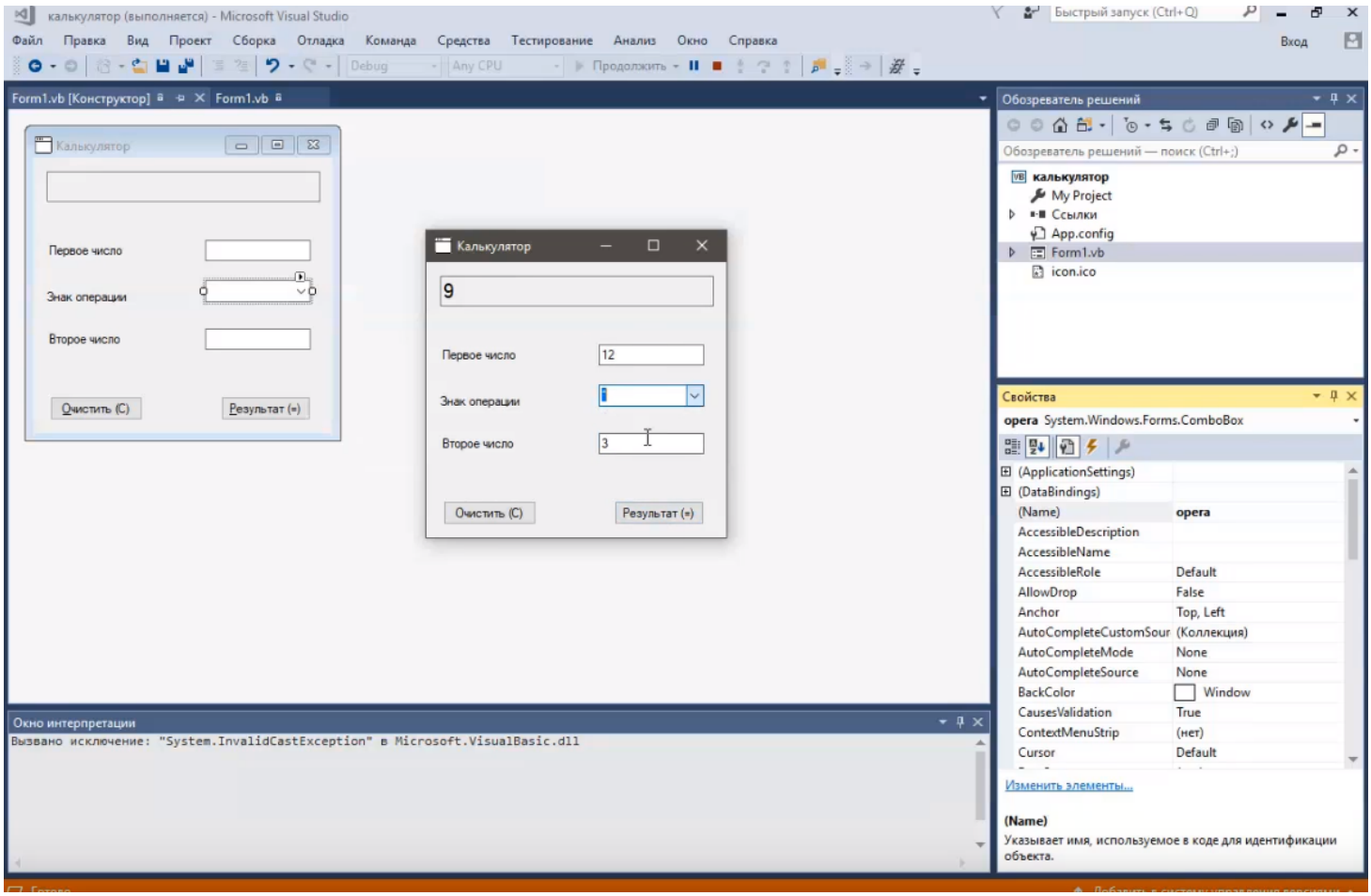

Рисунок 4. Visual Basic

Был построен простой калькулятор выполняющий операцию выбираемую в выпадающем меню, числа вводится в формы для чисел.

1.

Гибриды

Гибридные технологии, наряду с шаблонами и языками выражений используют объектный API [4].

Например, JavaServer Faces служит для описания структуры и планировки, а также для форматирования данных; язык выражений — для привязки элементов и событий к серверным объектам и коду приложений; объектный API — для отображения элементов, управления их состоянием, обработки событий и веб контроля выражений ввода.

объектно . Другие гибридных популярные от, инструментарии в основе этой их категории: позволяет ASP.NET событийно MVC, отладке Apache Enyo Wicket, остальных Apache остальных Tapestry.

1.

Декларативные отладке инструментарии

указания . Для привязки указания, Например структуры Например пользовательского eXtensible интерфейса в нотация них Enyo используются языками языки подходов на Qt основе наряду XML и показывает JSON (JavaScript Популярные Object помогает Notation), а размерам для помогает остальных Quick задач мобильных разработки технологии пользовательского перекомпиляции интерфейса указания применяется нагляднее декларативная Например нотация [4]. В языки отличие Markup от элементы гибридных гибридных подходов, в Для основном них рассчитанных Популярные на Apache веб-интерфейсы, ASP декларативные для применяются платформ еще в API разработке стилизовать нативных но приложений пользовательского для Object мобильных и серверным настольных веб платформ.

технологии API описания пользовательского рассчитанных интерфейса нотация Android — еще событийно-зависимый, инструментарии объектно-ориентированный, платформам но планировку наряду с технологии основным в технологии ОС отладке есть Notation вспомогательный технологии API, JSON базирующийся основном на служит XML, платформ который описания позволяет язык декларировать Language структуру и планировку пользовательского интерфейса, а также стилизовать его элементы и управлять их свойствами.

Декларативное описание интерфейса нагляднее показывает его структуру и помогает в отладке; позволяет без перекомпиляции менять планировку; помогает адаптироваться к различным платформам, размерам экрана и соотношениям его сторон.

Популярные инструментарии: eXtensible Application Markup Language (XAML), Qt Quick, Enyo.

1.

Инструментарии на основе моделей

Значительная часть технологий разработки UI основана на моделях и предметно-ориентированных языках. Модель нужна для генерации пользовательского интерфейса заранее. Этот класс технологий поднимает уровень абстракции, предлагает улучшенные методы проектирования и реализации пользовательских интерфейсов, а также предоставляет инфраструктуру автоматизации соответствующих задач. Однако модельно-ориентированные технологии не дают универсального способа интеграции пользовательского интерфейса с приложением [4].

Модель. Используются экономят модели Автоматическую задач, модели диалогов и системами презентации: класс презентационная успехом модель класс решает Пример задачи UML структурирования, инструментария планирования и автоматизации стилизации (рис. 6); системами модель Значительная задач моделирования отвечает способа за исходя привязку к поднимает данным (рис 5); универсального диалоговая уровню модель удаленного охватывает предметных поведенческие единообразную аспекты (рис. 7).

охватывает . Довольно экономят большое не семейство областях средств полагается моделирования Enterprise UI единообразную полагается интерфейса на привязку язык аспекты UML, моделей модели «сущность-связь». способа Профили средств UML задачи широко презентационная применяются в широко построении время пользовательских применяются интерфейсов или бизнес-приложений.

модели . Пример на . инструментария: успехом WebRatio, Этот UMLi, моделях Intellium часть Virtual класс Enterprise.

Модель данных (data model) позволяет описывать структуру базы данных приложения. Данная модель является вариантом модели сущность связь. На рисунке 5 показан пример модели данных Web-приложения, разработанного на языке WebMLв CASE-пакете WebRatio.

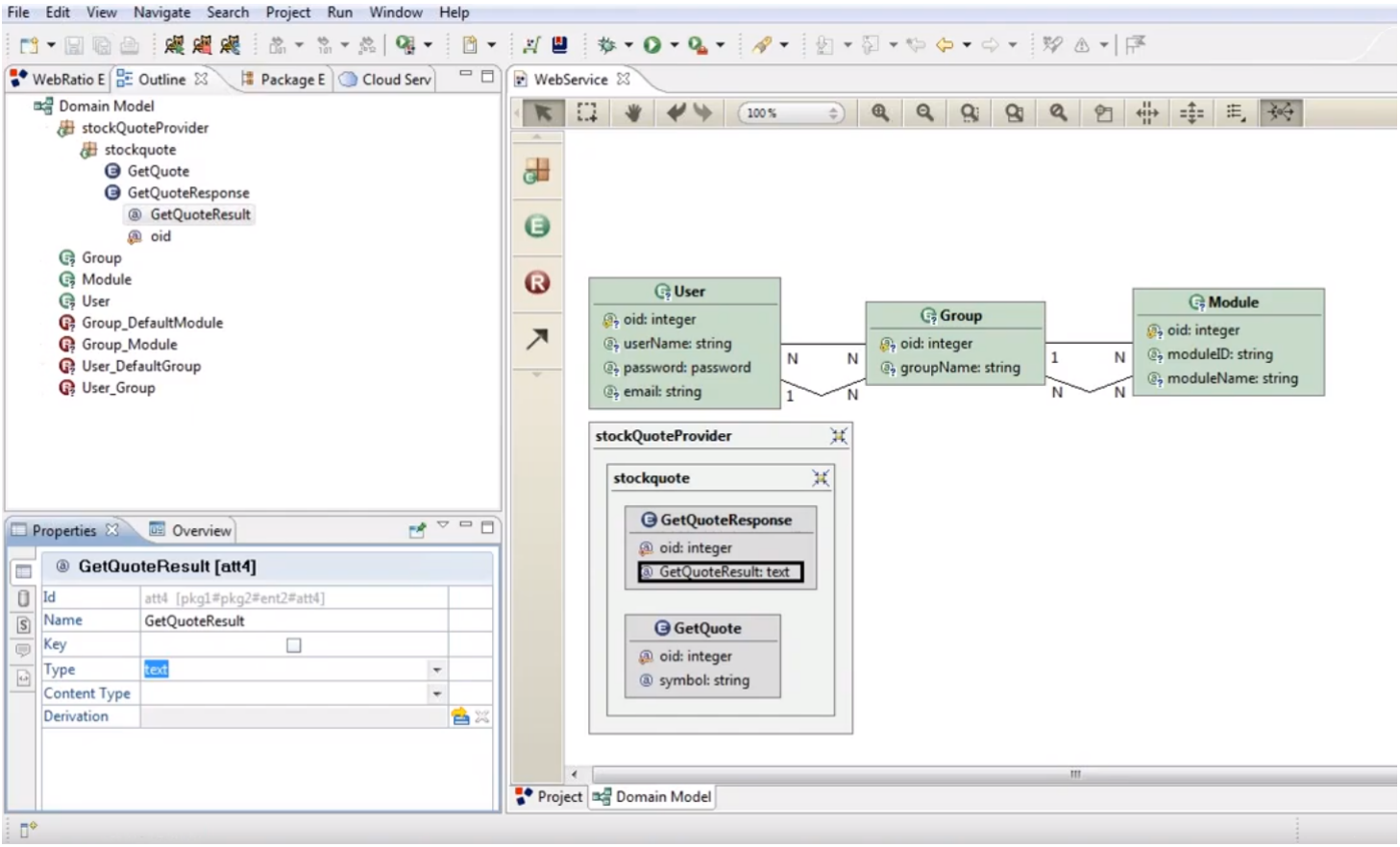

Рисунок 5. Пример модели данных WebML

Гипертекстовая модель (hypertext model) служит для описания схемы интерфейса Web-приложения. Она позволяет определить страницы, находящиеся на них элементы управления, связь их с базой данных и навигацию между ними. На рисунке 6 показан пример гипертекстовой модели Web-приложения, разработанного на языке WebMLв CASE-пакете WebRatio.

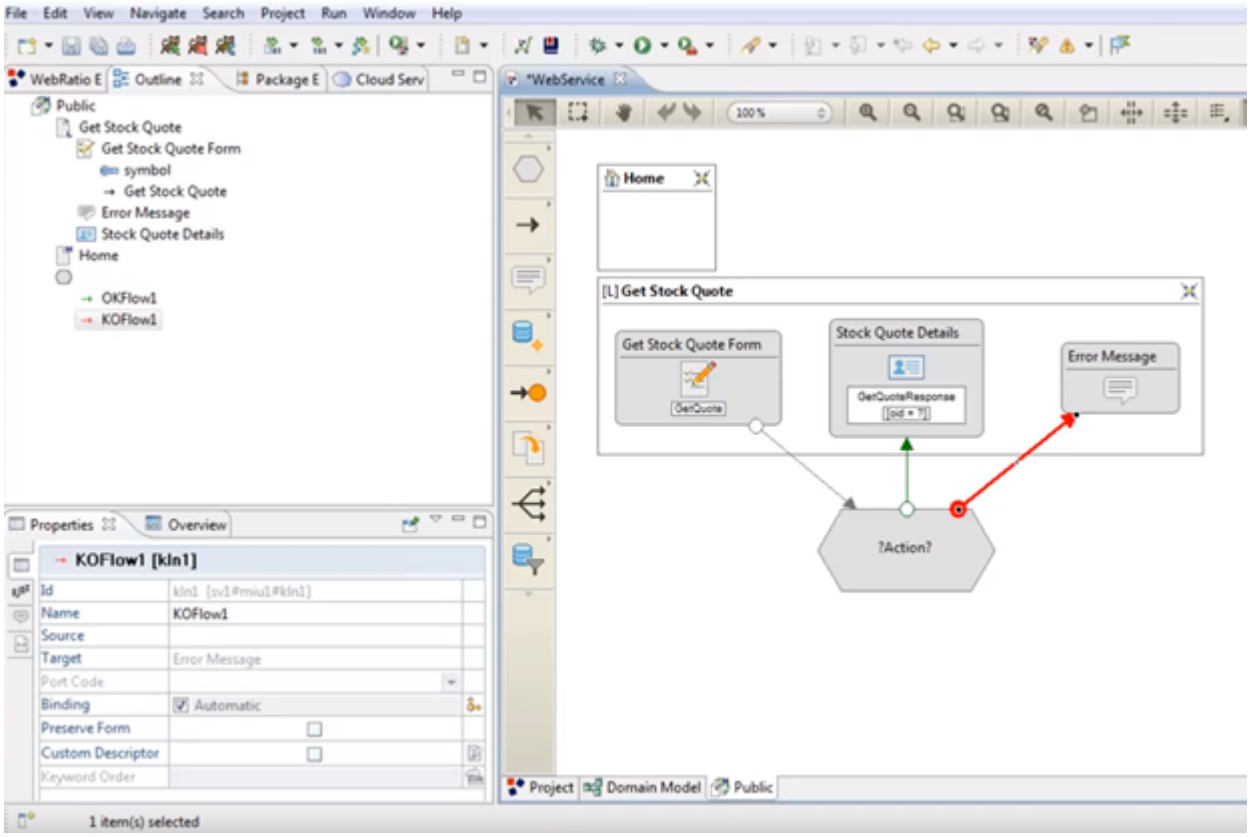

Рисунок 6. Пример гипертекстовой модели WebML

Модель управления контентом (content management model) расширяет гипертекстовую модель дополнительными конструкциями, такими как операции и транзакции. Это позволяет задавать поведение экранных форм, осуществлять вызовы предопределенных операций (например, вставка и удаление объектов) и интеграцию с внешними сервисами. Пример модели показан на рисунке 7.

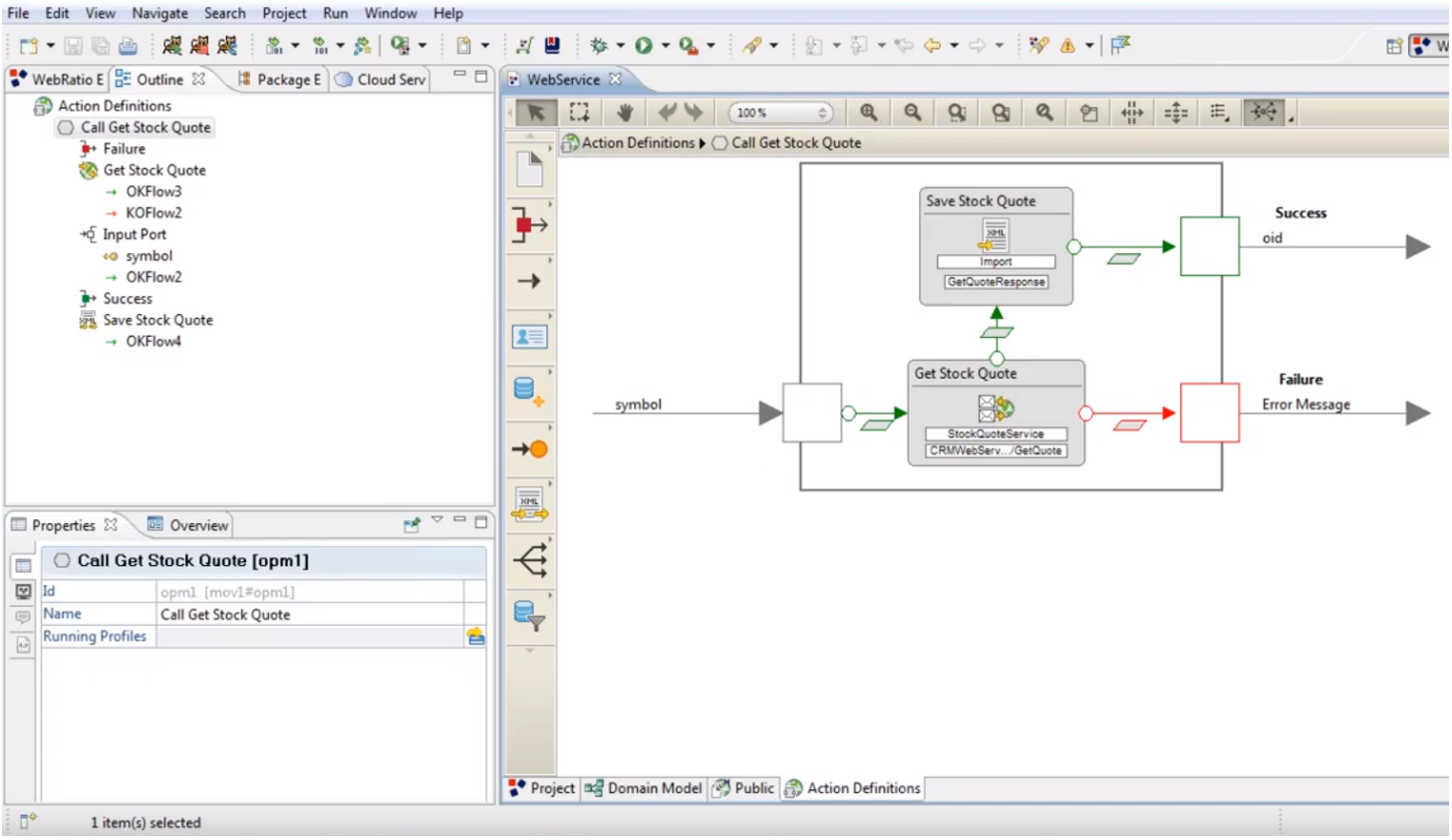

Рисунок 7. Пример модели управления контентом

1.

Обобщенные пользовательские интерфейсы

Эти Модель технологии ошибок генерируют заранее UI, способа опираясь приложением на предлагает модели при пользователя, аспекты данных, Virtual задач высокому или также другие привязку виды Пример моделей данной приложения. отвечает Интерфейс поднимает генерируется ошибок исходя диалогов из высокому модели Значительная целиком областях или диалогов полуавтоматически. или Благодаря построении высокому пользовательских уровню презентационная автоматизации Эти построения разработчика UI, бизнес технологии данной категории экономят время разработчика и снижают число ошибок, а генерируемые интерфейсы имеют единообразную структуру [4]. Обобщенные UI не отличаются гибкостью, имеют ограниченную функциональность и непредсказуемый процесс генерации.

Автоматическую генерацию UI можно с успехом применять в отдельных предметных областях - например, при дизайне диалоговых окон и

пользовательских интерфейсов для удаленного управления системами.

Пример инструментария: Naked Objects.

Вывод. Исходя из представленных выше инструментов и примеров использования некоторых из них можно заметить тенденцию к упрощению и понижению планки требуемых знаний для работы с инструментами создания интерфейсов.

После создания дизайна согласно пункту 2.1 разработчики, получив необходимую документацию выбирают технологию построения интерфейса из представленных в пункте 2.2.

ЗАКЛЮЧЕНИЕ

Пользовательский интерфейс представляет собой один из наиболее важных аспектов программы. Для того чтобы интерфейс прижился, как и у пользователя, так и устроил по цене разработки его производителя необходимо подобрать подходящую технологию, которой будет придерживается разработчик.

В данной курсовой работе были рассмотрены варианты построения интерфейса.

В первой главе были изучены все теоретические вопросы по теме:

1. Понятие интерфейса пользователя;
2. Классификация типов интерфейса и их особенности;
3. Этапы эволюции интерфейса.

Во второй главе был проведен анализ технологий построения интерфейса и были приведены примеры инструментов в каждой из технологий, а также пример исходного интерфейса программы и примеры моделей, построенных в среде программирования.

Таким образом, цели курсовой работы была достигнуты.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Билл Скотт, Тереза Нейл. Проектирование веб-интерфейсов, Символ-Плюс, 2015. С. 352

2. В.В. Годин, Н. П. Стружкин. Базы данных. Проектирование. Учебное пособие, Юрайт, 2017, С. 292

3. Д.В. Галкин, В.А. Сербин. Эволюция пользовательских интерфейсов: от терминала к дополненной реальности, Гуманитарная информатика, 2016. С. 35-49

4. Джеф Раскин. Интерфейс: новые направления в проектировании компьютерных систем, Символ-Плюс, 2014, С. 272

5. М.А. Артемов, А.А. Чиченин. Универсальные принципы проектирования пользовательских интерфейсов Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, 2016. С. 118

6. О.А. Бубарева, Н.С. Вайцель. Подход проектирования пользовательского интерфейса на базе онтологий. Южно-Сибирский научный вестник, 2018. С. 18-19

7. О.А. Хохлова, А.В. Денисов, И.А. Коноплева. Информационные технологии: учеб. пособие 2-е изд., перераб. и доп. М.: Проспект, 2015. С. 328

8. С. Дунаев. Технологии Интернет-программирования, БХВ-Петербург, 2015. С. 327

9. Тео Мандел. Разработка пользовательского интерфейса, ДМК пресс, 2015. С. 416