**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

Государственное образовательное учреждение высшего профессионального образования

**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”**

**им. В. И. Ульянова (Ленина)” (СПбГЭТУ)**

|  |
| --- |
|  |

Факультет КТИ УТВЕРЖДАЮ

Кафедра АСОИУ Зав. кафедрой Советов Б.Я.

Специальность 230102 «…….»………………………………

**ЗАДАНИЕ**

**на дипломное проектирование**

|  |
| --- |
| Студенту Кудрявцеву Ивану Владиславовичу |
| Место дипломного проектирования СПбГЭТУ «ЛЭТИ» |
|  |
| 1. Тема дипломного проекта (работы) |
| Разработка автоматизированной системы хранения программного кода и документации |
| для станков с ЧПУ |
|  |
| 2. Назначение разработки |
| Автоматизация хранения программ и документаций для станков с ЧПУ |
|  |
| 3. Перечень подлежащих разработке разделов по теме |
| Определения |
| Введение |
| Анализ предметной области и разработка технического задания |
| Программная реализация разработанной системы |
| Тестирование и подготовка презентации системы |
| Экономическое обоснование |
| Охрана интеллектуальной собственности |
|  |
| 4. Состав технической документации проекта |
| Пояснительная записка к дипломному проекту |
| Презентация |
|  |
| 5. Консультанты ( с указанием относящихся к ним разделов) |
| Консультант по экономическому обоснованию: Марков М. А. |
| Консультант по охране интеллектуальной собственности: Берковская К.И. |
|  |

Дата выдачи задания Дата сдачи проекта (работы)

«……»………………. «……»……………….

Руководитель………………………………………… Степулёнок Д.О.

Консультант от кафедры…………………………….. Степулёнок Д.О.

Студент……………………………………………….. Кудрявцев И.В.

**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

Государственное образовательное учреждение высшего профессионального образования

**“Санкт-Петербургский государственный электротехнический университет “ЛЭТИ”**

**им. В. И. Ульянова (Ленина)” (СПбГЭТУ)**

|  |
| --- |
|  |

Факультет      КТИ                УТВЕРЖДАЮ

Кафедра    АСОиУ               Зав. кафедрой     Советов Б.Я

Специальность     230102      «…….»………………………………

**КАЛЕНДАРНЫЙ ПЛАН**

**дипломного проектирования**

|  |
| --- |
| Студента **Кудрявцева Ивана Владиславовича** |
| Тема дипломного проекта (работы) |
| Разработка автоматизированной системы хранения программного кода и документации |
| для станков с ЧПУ |
| Срок дипломного проектирования: 4 месяца |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование раздела | Срок выполнения | Отметка  о выполнении |
| 1.  2.  3.  4.  5. | Анализ предметной области и разработка технического задания  Тестирование и подготовка презентации системы  Внедрение и тестирование  Технико-экономическое обоснование  Охрана интеллектуальной собственности | 19.10.11  20.12.11  05.01.12  18.01.12  25.01.12 | Выполнено  Выполнено  Выполнено  Выполнено  Выполнено |

Руководитель                                     Степулёнок Д.О.

Консультант от кафедры                  Степулёнок Д.О.

Студент                                                Кудрявцев И.В.

**Abstract**

The goal of this diploma work – is to develop automated storage system for documentation and program code for machines with numerical control.

System use MySQL database, PHP programming language (system is based on Yii Framework), HTML markup language, Cascading Style Sheets tables, JavaScript language.

Developed the database and web-interface.

Consolidation into a single database software code, drawings for it, as well as important information, provides a high efficiency of the equipment set-up and clarity of action.

System protected against unauthorized access to important files.

System realized foreign currency (revision control system), allowing convenient to organize, track and edit the files that have multiple versions.

Reduce the probability of errors and associated risk of equipment’s damage.

System can be modified and upgraded and implemented on a real production.

Оглавление

[Определения 6](#_Toc317176366)

[Введение 9](#_Toc317176367)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ 13](#_Toc317176368)

[1.1 Основные понятия и их определения 13](#_Toc317176369)

[1.2 Выбор средств реализации 16](#_Toc317176370)

[1.3 Языки и средства разметки для создания web-сайтов 19](#_Toc317176371)

[1.4 Языки программирования для создания серверной части 21](#_Toc317176372)

[1.5 Системы управления базами данных 24](#_Toc317176373)

[1.6 Yii Framework 27](#_Toc317176374)

[1.7 Средства и среды разработки 30](#_Toc317176375)

[1.8 Выводы 31](#_Toc317176376)

[1.9 Постановка задачи на дипломное проектирование 31](#_Toc317176377)

[2. РАЗРАБОТКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ 33](#_Toc317176378)

[2.1 Установка локального web-сервера denwer 33](#_Toc317176379)

[2.2 Настройка dbForge Studio для разработки 35](#_Toc317176380)

[2.3 Проектирование Базы Данных 36](#_Toc317176381)

[2.4 Общая структура проекта 38](#_Toc317176382)

[2.5 Модели для работы с БД 40](#_Toc317176383)

[2.6 Система авторизации 44](#_Toc317176384)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ СИСТЕМЫ 47](#_Toc317176385)

[3.1 Вход в систему 47](#_Toc317176386)

[3.2 Работа со списком программ 49](#_Toc317176387)

[3.3 Типы станков 50](#_Toc317176388)

[4. Технико-экономическое обоснование 52](#_Toc317176389)

[4.1 Концепция технико-экономического обоснования 52](#_Toc317176390)

[4.2 Потребительские свойства разрабатываемой научно-технической продукции. 53](#_Toc317176391)

[4.3 Анализ рынка и план маркетинга 54](#_Toc317176392)

[4.4 Производство продукта 55](#_Toc317176393)

[4.5 Расходы на оплату труда 56](#_Toc317176394)

[4.6 Отчисления на социальные нужды 57](#_Toc317176395)

[4.7 Расходы на материалы 57](#_Toc317176396)

[4.8 Накладные расходы 58](#_Toc317176397)

[4.9 Трудоемкость и календарный план 58](#_Toc317176398)

[4.10 Комплексная оценка эффективности работ 63](#_Toc317176399)

[5. ОХРАНА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ 67](#_Toc317176400)

[ПРИЛОЖЕНИЕ 85](#_Toc317176401)

[Выводы 99](#_Toc317176402)

[СПИСОК ЛИТЕРАТУРЫ 100](#_Toc317176403)

Определения

**ЧПУ (Числовое программное управление)** – компьютеризованная система управления, управляющая приводами металло- (помимо металлорежущих например, фрезерных или токарных), дерево- и пластмасообрабатывающих станков (центров), считывающая инструкции специализированного языка программирования, который затем интерпретатором системы ЧПУ переводится из входного языка в команды управления главным приводом, приводами подач, контроллерами управления узлов станка.

**Среда управляющей части станка с ЧПУ** - специализированное программное обеспечение, в которой разрабатываются программы для обработки деталей. Обладает своим интерфейсом и набором средств для управления отдельными узлами станка. Чаще является закрытой системой, т. е. системой со скрытым кодом. Среда управляющей части является основным средством управления станка.

**Главный привод** – двигатель, который вращает шпиндель.

**Привод подач** – двигатель, который передвигает деталь.

**Контроллер управления** – компьютер, который управляет работой всей системы.

**ISO 7-bit / G-код** — язык программирования устройств с числовым программным управлением (ЧПУ). Был создан компанией Electronic Industries Alliance в начале 1960-х. Финальная доработка была одобрена в феврале 1980 года как стандарт RS274D. Комитет ISO утвердил G-код, как стандарт ISO 6983-1:1982, Госкомитет по стандартам СССР — как ГОСТ 20999-83. В советской технической литературе G-код обозначается, как код ИСО 7-бит (ISO 7-bit).

***Пример****:*

*G00 Z1  
G00 X0 Y0  
M3 M7 F50  
G52 Z -0.1  
M98 P1  
G52 Z -.2  
M98 P1  
G52 Z -.3  
M98 P1  
G52 Z -.4  
M98 P1  
G52 Z -.5  
M98 P1  
G52 Z0  
M30*

**Управляющая часть станка –** компьютер, разработанный на базе процессоров “Intel Pentium”, с операционной системой “Microsoft Windows”. В управляющей части находится среда, посредством которой осуществляется управление станком.

**Программа для станка с ЧПУ** – средой разработки программ на станках “Monforts” является программное обеспечение “Sinumerik”, разработанное фирмой Siemens, в котором применяется язык программирования “ISO 7-bit” (G-code). Файлы с программами хранятся на жестких дисках управляющей части станка и имеют расширение “.mpf”. Такие файлы легко открываются любым текстовым редактором, поддерживающим расширение “.txt”.

**Документация для станков с ЧПУ** – набор чертежей и описаний для производства, которые объединены в технологический процесс. В технологических процессах описываются все действия в определенной последовательности для получения из заготовки конечного продукта.

**Клиент-сервер —** вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (сервисов), называемых серверами, и заказчиками услуг, называемых клиентами. Нередко клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. Программный код, обеспечивающий интерактивность web-страниц, называется сценарием (скриптом).

**Алгоритм** – совокупность операций, выполняемых в строго установленном порядке, для решения поставленной задачи.

**Несанкционированный доступ к информации (НСД) -** доступ к информации, нарушающий правила разграничения доступа с использованием штатных средств, предоставляемых средствами вычислительной техники или автоматизированными системами.

**Защита от несанкционированного доступа** - предотвращение или существенное затруднение несанкционированного доступа.

**Идентификация (обычно логин)** - присвоение субъектам и объектам доступа идентификатора и (или) сравнение предъявляемого идентификатора с перечнем присвоенных идентификаторов.

**Аутентификация (обычно пароль)** - проверка принадлежности субъекту доступа предъявленного им идентификатора, подтверждение подлинности.

**Целостность информации** - способность средства вычислительной техники или автоматизированной системы обеспечивать неизменность информации в условиях случайного и (или) преднамеренного искажения (разрушения).

**MVC (Model-View-Controller)** – шаблон проектирования приложений Модель-Представление-Контроллер. Шаблон проектирования, при котором модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Введение

На производстве, где используется современное оборудование, зачастую наблюдаются случаи неполной реализации потенциала доступных технологических ресурсов. Это влечет за собой потери эффективности использования высокотехнологичного оборудования.

На примере ОАО ”Кировский завод” можно увидеть утечки производственных мощностей из-за различных недочетов в процессах производства. Одним из таких случаев является отсутствие автоматизации при наладке станков и отсутствие какого-либо технического мониторинга за производительностью оборудования.

В данной дипломной работе рассматриваются немецкие металлообрабатывающие станки с числовым программным управлением (далее ЧПУ) ”Monforts”. Станки “Monforts” оснащены компьютерами, которые имеют возможность коммутировать между собой и с другими системами посредством TCP/IP соединения. Это дает возможность создать из станков с ЧПУ компьютерную сеть, во главе с сервером, на котором находилось бы программное обеспечение, позволяющее более быстро и эффективно настраивать станки.

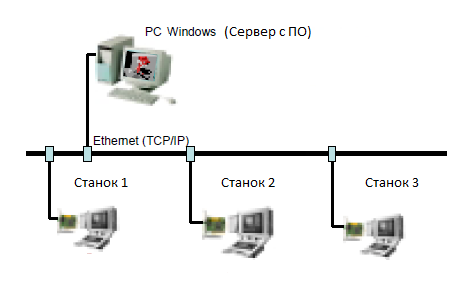


Рис. 1. Схема компьютерной сети

В настоящее время процесс наладки станка занимает много времени, вследствие того, что необходимая для производства той или иной детали документация не организованна и разбросана по производственному участку, а соответствующую этой документации программу приходится искать на разных станках.

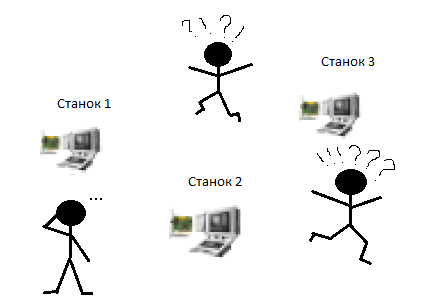


Рис. 2. Потеря эффективности производства

Большинство программ имеют несколько версий, т. к. могут быть предназначены для разных станков. Зачастую, практически невозможно определить принадлежность программы к тому или иному станку, тщательным образом не разобравшись в её коде, что ведет к большим затратам времени. Неверно запущенные на станках программы, ведут к возникновению ошибок в процессе наладки и вызывают довольно-таки весомый материальный ущерб. Неверно запущенная программа может привести к поломке дорогих элементов станка, сломанному инструменту и производству брака. Также, неправильно работающий станок является повышенным источником опасности для здоровья и жизни человека.

Вполне очевидно, что отсутствие какой-либо системы, которая обеспечивала бы порядок в процессе наладки, приводит к затрате большого количества времени и усилий, а также является причиной ошибок, которые могут привести к тяжелым последствиям. Это естественным образом вызывает потери в доходах того или иного предприятия, т.к. в процессе наладки станки не работают, а за допущенные из-за недостаточной производственной дисциплины ошибки приходится дорого платить.

Данная дипломная работа посвящена разработке автоматизированной системы для устранения описанных проблем.

Рис. 3. Станок с ЧПУ «Monforts»

Monforts (полное название A. Monforts erkzeugmaschinen GmbH & Co. KG) — немецкая компания, изготавливающая высококачественную продукцию в области станкостроения и текстильного оборудования.

Преимущество фирмы Monforts — серийное производство токарных станков и токарных обрабатывающих центров с ЧПУ с цилиндрическими гидростатическими направляющими. На станках могут обрабатываться детали длиной от 250 мм до 1500 мм и диаметром от 300 мм до 1100 мм. Большой акцент фирма делает на безопасность труда, легкий доступ к рабочей зоне станка, наглядность панели управления.



Рис. 4. Токарно-фрезерный центр Monforts UniCen 1000 MultiTurn

Модельный ряд станков с ЧПУ:

* RNC - высокопроизводительные токарные станки с ЧПУ
* MHC - горизонтальные станки Pick-up
* DNC - токарные станки с ЧПУ и противошпинделем
* MNC - модульные высокопроизводительные токарные станки с ЧПУ
* UniCen - модульные высокопроизводительные токарно-фрезерные центры с ЧПУ

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ
   1. Основные понятия и их определения

Для работы на станочном оборудовании, в особенности с ЧПУ, необходимо сначала настроить его, т. е. произвести *процесс наладки*. В настоящее время, процесс наладки станочного оборудования, прежде всего, включает в себя поиск документации и нужной программы для производства детали. После того, как программа и соответствующая ей документация найдены, осуществляется загрузка программы в управляющую часть станка.

Под *управляющей частью* станка понимают компьютеризованную систему управления, управляющую приводами станков (центров), считывающую инструкции специализированного языка программирования, который затем интерпретатором системы ЧПУ переводится из входного языка в команды управления главным приводом, приводами подач и контроллерами управления узлов станка.

Далее, после загрузки программы в управляющую часть, происходит последующая настройка инструмента. Например:

* определение типа и параметров зажима в патроне;
* измерение длины инструмента;
* привязка к нулям детали;
* определение положения инструмента в револьверной головке;
* определение положения револьверной головки относительно осей станка;

Рис. 5. Процесс наладки рабочей части станка.

Настройка инструмента выполняется при помощи среды, установленной на управляющую часть станка. В данной работе рассматривается среда Siemens Sinumerik.

Siemens Sinumerik – полностью цифровая система для практически всех типов применений. Это системная платформа с прогрессивными функциями. Sinumerik представляет собою систему, которая подходит для сложных задач обработки и демонстрирует высокий уровень динамики и точности. Во всем мире Sinumerik применяется для: токарной обработки, сверления, фрезерования, шлифования, лазерной обработки, порезки, перфорации, изготовления оснастки и инструмента, как система управления прессами, для высокоскоростного раскроя материалов, обработки древесины и стекла, транспортировки, складских задач.

Sinumeric в модуле NCU (Numeric Control Unit — устройство числового управления) объединяет задачи ЧПУ, ПЛК и коммуникации. Установленный в каркас-носитель, NCU встраивается непосредственно в цифровую систему преобразования SIMODRIVE 611D, при этом он располагается справа, непосредственно у модуля питания-рекуперации.

Варианты процессоров NCU и системное программное обеспечение дает возможность оптимальной адаптации к станку и к задаче обработки. Такой модульный принцип позволяет оснастить целый ряд станков различного типа.

При помощи SINUMERIK 840D можно управлять максимум 31 осями/шпинделями. При максимальном использовании поддерживается до 10 каналов на каждую группу режимов работы и максимум 12 осей/шпинделей на каждый канал. Каждый канал может иметь свою собственную группу режимов работы.

SINUMERIK 840D позволяет просто и экономично обеспечить высокоэффективную защиту обслуживающего персонала и станков благодаря встроенным сертифицированным функциям защиты.

Все NCU изначально имеют встроенное подключение 4 быстрых цифровых входов/выходов ЧПУ.

Возможно объединение нескольких систем управления в одну.

После проведения настройки инструмента, наладчик приступает к пробной обработке детали. После того, как все эти этапы прошли успешно, по необходимости производится корректировка размеров и, станок становится полностью готовым к работе. На выполнение этих действий, как правило, затрачивается неопределенное количество времени, которое невозможно каким-либо образом предугадать. Это происходит вследствие некоторых причин, независящих от наладчика. Например:

* затрудненный поиск технологических процессов и программ для станков с ЧПУ
* отсутствие необходимого инструмента (кулачков, оправок, пластинок и пр.)
* наличие внешних факторов (отключение электричества, производственное собрание)
* неожиданная поломка оборудования, обеспечивающего работу станков (поломка гидравлической системы или конвейера для стружки)

Данный дипломный проект направлен, прежде всего, на устранение первого пункта из вышеизложенного списка. Для устранения данной проблемы необходимо создать базу данных с удобным интерфейсом, низкими системными требованиями и возможностью работы с ней из любого места, где есть подключение к интернету.

* 1. Выбор средств реализации

Архитектура приложения состоит из трех слоев:

* Клиентский (Браузер)
* Серверный (Web-сервер)
* База данных

База Данных с документами и программным кодом

Серверное приложение

Клиентская часть (интерфейс)

Хранение данных

Разграничение доступа,

защита информации

Пользовательский интерфейс

Рис. 6. Клиент-серверная архитектура системы

**Варианты реализации:**

* **Web-приложение**

Преимущества: не требуется установка (простота использования), централизованное обновление, обеспечение защиты, низкие требования к ресурсам компьютеров клиентов; современные технологии программирования постоянно развиваются в этом направлении, давая программистам все новые и новые инструменты по разработке именно web-ориентированных приложений; в обычных настольных приложениях чаще всего сервер базы данных не используется, а используется файловая модель хранения данных (например, в файлах Microsoft Access или DBF). Напротив, с большой долей вероятности, используя web-приложение, данные будут храниться в специальном SQL-сервере (так удобнее провайдерам). То есть надежность хранения данных у типичного web-приложения будет выше, чем у настольного приложения. Отсюда вытекает еще ряд достоинств использования web-приложения (из области сервисных задач): реализация разграничения доступа и резервного хранения информации обычно решена изначально в рамках типовых решений провайдера. Причем эти решения отработаны и оттестированы на многочисленных пользователях, что повышает степень доверия к ним и их надежность; легко модернизируема; можно получить доступ к нужным файлам из любого места, где есть компьютер с подключением к интернету.

Недостатки: скорость работы (для цели данной работы этот недостаток является несущественным), гибкость, затруднено использование внешних медиа-файлов (видео), требуется наличие постоянного соединения с интернетом.

* **Обычное настольное приложение**

Преимущества: возможность создания более мощного интерфейса.

Недостатки: массивность; низкая надежность хранения данных; для доступа к файлам, каждый раз необходимо будет устанавливать приложение туда, где оно будет использоваться (в данной работе этот недостаток является неприемлемым), низкая степень защиты информации;

* **Приложение для мобильных платформ**

Преимущества: малый размер; возможность использования такого приложения в любое время и в любом месте; легко в использовании;

Недостатки: низкая функциональность (недостаточная для реализации данного проекта); низкая степень надежности; отсутствие кроссплатформенности; неудобства средств управления приложением (отсутствие мыши, клавиатуры и пр.)

Для выполнения данного проекта, была выбрана реализация в виде Web-приложения. Основными предпосылками к этому стали:

* возможность использования такого приложения из любого места, где есть компьютер и подключение к интернету, т.к. зачастую то или иное производство разделено территориально (находится в разных цехах, производственных участках), но, при этом, имеет одни и те же задачи;

*К примеру: два одинаковых станка “а” и “б”, которые надо настроить на одну и ту же деталь располагаются в различных цехах: цехе ВХ 28 и в цехе №320, расстояние между которыми примерно равно 0.7 км. На станке “а” программа есть, а на станке “б” нет. Для того что бы настроить станок “б”, наладчику будет необходимо с каким-либо внешним накопителем преодолеть расстояние между цехами, скопировать программу, тем самым приостановив работу станка “а” и вернуться обратно. На выполнение данных действий он затратит от получаса времени. А при наличии доступа к единой базе данных это время сократится примерно в 6 раз.*

* простота использования, обусловленная тем, что web-приложение не требует установки;
* наличие централизованного сервера – это дает возможность объединить в единой базе данных необходимые для работы файлы с программами и документацией, структурировать их и осуществлять быстрый поиск, что позволяет экономить массу время и предотвратить возникновение ошибок, которые являются следствием “неразберихи” в процессе наладки оборудования.

Также, такие качества как надежность, низкие требования к компьютерным ресурсам, возможность модернизации, наличие централизованного обновления, высокая степень защиты и постоянное развитие в программировании Web-приложений, делают данный способ реализации актуальным и востребованным. Недостатки Web-приложений для выполнения данной работы являются несущественными.

* 1. Языки и средства разметки для создания web-сайтов

**HTML**, Hyper Text Markup Language (язык разметки гипертекста) — базовая технология интернета. HTML - стандартный язык разметки документов во Всемирной паутине. Большинство web-страницы создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузером и отображается в виде документа, в удобной для человека форме.

При помощи HTML можно создавать отдельные элементы, а в дальнейшем компоновать их между собой. HTML позволяет формировать текстовые блоки, форматировать их, выделяя абзацы, организовывать и создавать любые таблицы и списки. Есть возможность управлять цветовым оформлением документов, добавлять на сайт звуки, анимацию, картинки. HTML, естественно, помогает организовывать связь документов, разделов и страниц сайта между собой посредством гиперссылок.

Главный недостаток HTML — отсутствие средств, при помощи которых было бы возможно контролировать просмотр web-страниц независимо от пользователя. Web-мастер подготавливает страницу под какие-то определенные параметры, разрешение монитора, вид браузера и т.д. Пользователь же, изменив эти параметры у себя на компьютере, может получить на выходе страницу, существенно отличающуюся от замыслов программиста.

**XML** (eXtensible Markup Language, расширяемый язык разметки) - язык структурной разметки текстовой информации. XML предназначен для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки (например, XHTML), иногда называемых словарями. XML является упрощённым подмножеством языка SGML.

Главное преимущество XML заключается в том, что язык этот можно дописывать и расширять, добавляя новые тэги, создавая свои пользовательские XML-словари. По мнению аналитиков, XML постепенно должен стать основным стандартом во всемирной сети, включив в себя большинство других использующихся ныне форматов.

**CSS**. Технология CSS (Cascading Style Sheets, каскадные таблицы стилей) позволяет с помощью специального макроязыка задать форматирование страницы. В файле CSS мы можем задать параметры для всех тегов HTML по отдельности, а также для группы тегов, создать некий шаблон, данные которого будут в дальнейшем влиять на все элементы страницы.

Стили можно задать не только для одной конкретной страницы, но и для всего сайта в целом. Также технологию CSS можно использовать практически на любом сервере без каких-либо ограничений.

**DHTML** (Dynamic Hyper Text Markup Language) – это способ создания интерактивного веб-сайта, использующий сочетание статичного языка разметки HTML, встраиваемого (и выполняемого на стороне клиента) скриптового языка JavaScript и CSS (каскадных таблиц стилей).

DHTML базируется на объектной модели документа DOM (Document Object Model), технология эта встроена во все виды браузеров и помогает осуществить доступ к любому отдельно взятому элементу web-страницы (который в дальнейшем можно видоизменить). Технология DHTML позволяет нам осуществлять программное управление всеми тегами и атрибутами и создавать сложные пользовательские интерфейсы. DHTML позволит программисту спроектировать такие интерактивные элементы, как движущийся фон, расположенный под статичным текстовым содержимым страницы, любые сложные объекты, выпадающие меню, мигающие при наведении курсора кнопки, анимацию, титры и многое другое.

* 1. Языки программирования для создания серверной части

Все описываемые языки программирования относятся к числу объектно-ориентированных.

Из них Perl, РНР, Python, JAVA — языки, исполняющиеся на сервере (серверные языки). JavaScript — язык, исполняющийся на браузере (клиентский). Компьютер, открывающий доступ к собственным ресурсам, носит название **сервера**, а получающий такой доступ — **клиент**.

**Perl** (Practical Extrifiction and Report Language, практический язык выборки и составления отчетов) оптимизирован для обработки большого количества текстовых файлов. Язык Perl способен обрабатывать огромное количество документов на web-узле (сервере), многие серверы написаны как раз на языке Perl.

Web-мастеру Perl поможет проверить верность перекрестных ссылок на сайтах, осуществит проверку заполнения элементов формы, позволит присоединять базы данных с последующей их обработкой. Perl может работать и с протоколом FTP. Это позволяет автоматизировать получение файлов с других серверов, а в сочетании с возможностями языка по обработке текстовых файлов дает web-мастеру возможность создавать сложные информационные системы.

**PHP–** это скрипт-язык (scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере.

Язык программирования PHP создавался специально для сети. PHP лучше всего охарактеризовать так: программист пишет сценарии — код, сервер, получая готовые указания, обрабатывает информацию и выводит результат на монитор пользователя.

PHP помогает создавать динамические web-приложения, такие как форумы, счетчики статистики, может работать с большими объемами данных — массивами. Очень хорошо язык PHP подходит для написания систем управления сайтами «движков».

Для того чтобы сайт быстрее загружался, стоит разбить страницу на отдельные элементы. Язык программирования PHP позволяет «на лету», за считанные доли секунды, объединять отдельные составляющие в цельную web-страницу.

Основное достоинство языка PHP — простота изучения. Он будет интуитивно понятен программистам, обладающим навыками в таких языках, как Perl и C, и для новичка вполне доступен.

В PHP нет поддержки ссылок на переменные, из-за чего сложно обрабатывать масштабные структуры данных, например, вложенные списки. Нет проверки типов данных передаваемых функций, что может повлечь за собой множественные ошибки.

Основное отличие от CGI-скриптов, написанных на других языках, типа Perl или C – это то, что в CGI-программах вы сами пишете выводимый HTML-код, а, используя PHP – вы встраиваете свою программу в готовую HTML-страницу, используя открывающий и закрывающий теги (в примере <?php и ?>).

Отличие PHP от JavaScript, состоит в том, что PHP-скрипт выполняется на сервере, а клиенту передается результат работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется.

**Python** прост для освоения, а между тем этот язык программирования на уровень выше, чем PHP и C. Это достигается за счет встроенных высокоуровневых структур данных (списки, массивы, словари).

Главное его достоинство то, что интерпретатор Python реализован практически на всех платформах и операционных системах.

**JAVA** представлен на сегодняшний момент в двух вариантах: JavaScript и собственно JAVA.

**JavaScript** — по сути, всего лишь надстройка стандарта HTML, которая значительно расширяет возможности документа, созданного в этом формате. Часть кода, написанная на JavaScript, внедряется в документ HTML как подпрограмма и вызывается на исполнение соответствующей строкой HTML-кода. Весь сценарий JavaScript умещается в одном-единственном теге <script>. Встроенный в браузер интерпретатор языка воспринимает и скрипт, и сам код гипертекста как единый документ, обрабатывая те и другие данные одновременно.

При помощи JavaScript можно полностью контролировать формат всплывающих окон и встроенных фреймов, организовывать такие интерактивные элементы, как часы, календарь, бегущие строки и иную анимацию, создать чат, сделать красивое меняющееся меню.

**JAVA** — это уже нечто большее: полноценный кроссплатформенный язык программирования, который будет правильно обработан и компьютером с операционной системой Windows, и ПК под управлением UNIX. Большинство web-камер, передающих на сайты изображение в реальном времени, работают на базе приложений JAVA.

**ASP** (Active Server Pages, активные страницы сервера) — это мощная технология от компании Microsoft, позволяющая легко разрабатывать интернет-приложения и служащая для создания динамических web-страниц. ASP нельзя назвать языком программирования, так как это только технология предварительной обработки, позволяющая подключать программные модули во время процесса формирования страницы. Ее можно сравнить с библиотекой для сети: «Большой энциклопедический словарь» с возможностью поиска по архивам. ASP имеет преимущество в скорости по сравнению с другими технологиями, основанными на скриптах.

Удобство ASP заключается в том, что она работает на специальном сервере PWS (Personal Web Server), который входит в состав операционной системы Windows.

* 1. Системы управления базами данных

**MySQL –** это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет-сайтов, как небольших, так и достаточно крупных.

MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей. Поддержка сервера MySQL автоматически включается в поставку PHP.

Немаловажным фактором является ее бесплатность. MySQL распространяется на условиях общей лицензии GNU (GPL, GNU Public License).

Ранее для долговременного хранения информации мы работали с файлами: помещали в них некоторое количество строчек, а затем извлекали их для последующей работы. Задача длительного хранения информации очень часто встречается в программировании Web-приложений: подсчёт посетителей в счётчике, хранение сообщений в форуме, удалённое управление содержанием информации на сайте и т.д.

Между тем, профессиональные приёмы работы с файлами очень трудоёмки: необходимо заботится о помещении в них информации, о её сортировке, извлечении, при этом не нужно забывать, что все эти действия будут происходить на сервере хост-провайдера, где с очень большой вероятностью стоит один из вариантов Unix - следовательно, нужно так же заботится о правах доступа к файлам и их размещении. При этом объем, кода значительно возрастает, и совершить ошибку в программе очень просто.

Все эти проблемы решает использование базы данных. Базы данных сами заботятся о безопасности информации и её сортировке, и позволяют извлекать и размещать информацию при помощи одной строчки. Код с использованием базы данных получается более компактным, и отлаживать его гораздо легче. Кроме того, не нужно забывать и о скорости - выборка информации из базы данных происходит значительно быстрее, чем из файлов.

Приложение на РНР, использующее для хранения информации базу данных (в частности MySql) всегда работает быстрее приложения, построенного на файлах. Дело в том, что базы данных написаны на языке C++, и написать на PHP программу, которая работала бы с жёстким диском эффективнее базы данных - задача неразрешимая по определению, поскольку программы на PHP в принципе работают медленнее, чем программы на C++, так как РНР - интерпретатор, а С++ компилятор.

Таким образом, основное достоинство базы данных заключается в том, что она берёт на себя всю работу с жёстким диском и делает это очень эффективно.

* **Таблицы «ключ-значение» (хэш-таблицы)**

Это [структура данных](http://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), реализующая интерфейс [ассоциативного массива](http://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%81%D0%BE%D1%86%D0%B8%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2), а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу. Существует два основных варианта хеш-таблиц: с цепочками и открытой адресацией. Хеш-таблица содержит некоторый массив H, элементы которого есть пары (хеш-таблица с открытой адресацией) или списки пар (хеш-таблица с цепочками).

Выполнение операции в хеш-таблице начинается с вычисления [хеш-функции](http://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) от ключа. Получающееся хеш-значение i = hash(key) играет роль индекса в массиве H. Затем выполняемая операция (добавление, удаление или поиск) перенаправляется объекту, который хранится в соответствующей ячейке массива H[i].

Ситуация, когда для различных ключей получается одно и то же хеш-значение, называется [коллизией](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). Такие события не так уж и редки — например, при вставке в хеш-таблицу размером 365 ячеек всего лишь 23-х элементов вероятность коллизии уже превысит 50 % (если каждый элемент может равновероятно попасть в любую ячейку) . Поэтому механизм разрешения коллизий — важная составляющая любой хеш-таблицы.

В некоторых специальных случаях удаётся избежать коллизий вообще. Например, если все ключи элементов известны заранее (или очень редко меняются), то для них можно найти некоторую [совершенную хеш-функцию](http://ru.wikipedia.org/w/index.php?title=%D0%A1%D0%BE%D0%B2%D0%B5%D1%80%D1%88%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F&action=edit&redlink=1), которая распределит их по ячейкам хеш-таблицы без коллизий. Хеш-таблицы, использующие подобные хеш-функции, не нуждаются в механизме разрешения коллизий, и называются хеш-таблицами с прямой адресацией.

Число хранимых элементов, делённое на размер массива H (число возможных значений хеш-функции), называется коэффициентом заполнения хеш-таблицы (load factor) и является важным параметром, от которого зависит среднее время выполнения операций.

* **Goggle App Engine**

Google App Engine позволяет создавать и размещать веб-приложения в тех же системах, в которых находятся приложения Google. В App Engine можно быстро и легко разработать, развернуть, администрировать и масштабировать приложения, не заботясь об аппаратных средствах, исправлениях и резервном копировании.

Разработанные в этой среде приложения являются зависимыми от компании “Google”, что является неприемлемым для данной работы.

* 1. Yii Framework

Yii – это высокоэффективный основанный на компонентной структуре PHP-фреймворк для разработки масштабных веб-приложений. Он позволяет максимально применить концепцию повторного использования кода и может существенно ускорить процесс веб-разработки. Название Yii (произносится как Yee) означает простой (easy), эффективный (efficient) и расширяемый (extensible).

Yii использует шаблон проектирования Модель-Представление-Контроллер (MVC, Model-View-Controller), который широко применяется в веб-программировании.

MVC направлен на отделение бизнес-логики от пользовательского интерфейса, чтобы разработчики могли легко изменять отдельные части приложения не затрагивая другие. В архитектуре MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс (например, текст, поля ввода), а контроллер обеспечивает взаимодействие между моделью и представлением.

Помимо этого, Yii использует фронт-контроллер, называемый приложением (application), который инкапсулирует контекст обработки запроса. Приложение собирает информацию о запросе и передает его для дальнейшей обработки соответствующему контроллеру.

Следующая диаграмма отображает структуру приложения Yii:

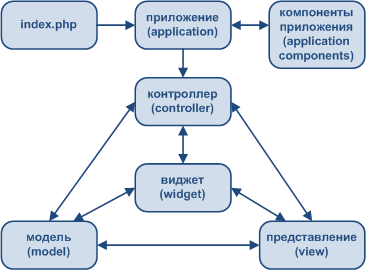


Рис. 7. Структура приложения Yii

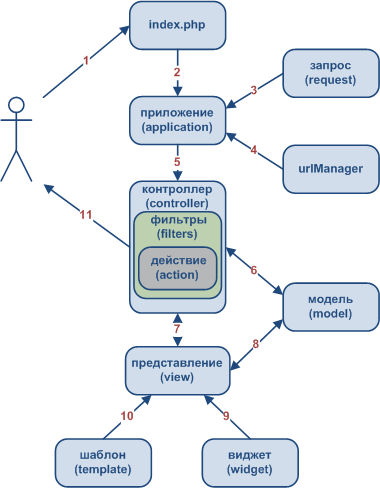


Рис. 8. Типичная последовательность работы приложения Yii

1. Пользователь осуществляет запрос посредством URL **http://www.example.com/index.php?r=post/show&id=1**, а веб-сервер обрабатывает его, запуская выполнение скрипта инициализации **index.php**;
2. Скрипт инициализации создает экземпляр [приложения](http://yiiframework.ru/doc/guide/ru/basics.application) и запускает его на выполнение;
3. Приложение получает подробную информацию о запросе пользователя от [компонента приложения](http://yiiframework.ru/doc/guide/ru/basics.application#application-component) **request**;
4. Приложение определяет запрошенные [контроллер](http://yiiframework.ru/doc/guide/ru/basics.controller) и [действие](http://yiiframework.ru/doc/guide/ru/basics.controller#action) при помощи компонента **urlManager**. В данном примере контроллером будет **post**, относящийся к классу **PostController**, а действием —**show**, суть которого определяется контроллером;
5. Приложение создает экземпляр запрашиваемого контроллера для дальнейшей обработки запроса пользователя. Контроллер определяет соответствие действия **show** методу **actionShow** в классе контроллера. Далее создаются и применяются фильтры (например, access control, benchmarking), связанные с данным действием, и, если фильтры позволяют, действие выполняется;
6. Действие считывает из базы данных [модель](http://yiiframework.ru/doc/guide/ru/basics.model) **Post** с ID равным **1**;
7. Действие формирует [представление](http://yiiframework.ru/doc/guide/ru/basics.view) **show** с данными модели **Post**;
8. Представление получает и отображает атрибуты модели **Post**;
9. Представление выполняет некоторые [виджеты](http://yiiframework.ru/doc/guide/ru/basics.view" \l "widget);
10. Сформированное представление добавляется в [макет страницы](http://yiiframework.ru/doc/guide/ru/basics.view#layout);
11. Действие завершает формирование представления и выводит результат пользователю.
    1. Системы контроля версии и возможности их применения

**Система управления версиями** (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Такие системы наиболее широко применяются при разработке программного обеспечения, для хранения исходных кодов разрабатываемой программы. Однако, они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов, в частности, они применяются в САПР, обычно, в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools). Ситуация, когда электронный документ за время своего существования претерпевает ряд изменений, достаточно типична. При этом часто бывает важно иметь не только последнюю версию, но и несколько предыдущих. В простейшем случае, можно просто хранить несколько вариантов документа, соответствующим образом их нумеруя. Но такой способ неэффективен (приходится хранить несколько практически идентичных копий), требует повышенного внимания и дисциплины и часто ведёт к ошибкам. Поэтому были разработаны средства для автоматизации этой работы.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет бо́льшую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы (например, ClearCase) поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

Иногда создание новой версии выполняется незаметно для пользователя (прозрачно), либо прикладной программой, имеющей встроенную поддержку такой функции, либо за счёт использования специальной файловой системы. В этом случае пользователь просто работает с файлом, как обычно, и при сохранении файла автоматически создаётся новая версия.

Часто бывает, что над одним проектом одновременно работают несколько человек. Если два человека изменяют один и тот же файл, то один из них может случайно отменить изменения, сделанные другим. Системы управления версиями отслеживают такие конфликты и предлагают средства их решения. Большинство систем может автоматически объединить (слить) изменения, сделанные разными разработчиками. Однако такое автоматическое объединение изменений, обычно, возможно только для текстовых файлов и при условии, что изменялись разные (непересекающиеся) части этого файла. Такое ограничение связано с тем, что большинство систем управления версиями ориентированы на поддержку процесса разработки программного обеспечения, а исходные коды программ хранятся в текстовых файлах. Если автоматическое объединение выполнить не удалось, система может предложить решить проблему вручную.

Часто выполнить слияние невозможно ни в автоматическом, ни в ручном режиме, например, если формат файла неизвестен или слишком сложен. Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла (например, средствами файловой системы) и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

Многие системы управления версиями предоставляют ряд других возможностей:

* Позволяют создавать разные варианты одного документа, т. н. ветки, с общей историей изменений до точки ветвления и с разными — после неё.
* Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.
* Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.
* Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.
  1. Средства и среды разработки

**Работа с базой данных MySQL**

dbForge Studio for MySQL

Страница продукта:

<http://www.devart.com/ru/dbforge/mysql/studio/download.html>

**HTTP-сервер для разработки**

Для разработки использовался локальный сервер из пакета Denwer (Джентльменский Набор Web-Разработчика). <http://www.denwer.ru/>

В пакет denwer входят:

* HTTP-сервер Apache;
* Интерпретатор языка программирования PHP;
* Сервер баз данных MySQL.

**Среда web-разработки для PHP/HTML/CSS/JavaScript PHPStorm.**

<http://www.jetbrains.com/phpstorm/>

PHPStorm – среда для разработки на PHP от компании JetBrains.

* 1. Выводы

В этой главе мы рассмотрели общие сведения, связанные со станками с ЧПУ и программированием для них. Для повышения эффективности работы операторов-наладчиков станков необходимо систематизировать документацию и программный код станков.

1. Представлены основные понятия и определения Web-сайта.
2. Описана классификация атак на Web-сайт и возможные средства защиты от них.

Из выполненного аналитического обзора вытекает следующая актуальная задача дипломного проекта: комплексная система защиты сайта.

* 1. Постановка задачи на дипломное проектирование

Тема разработки: автоматизированная система хранения программного кода и документации для станков с ЧПУ.

Система будет реализована в виде web-системы, которая будет установлена на сервере локальной сети предприятия.

Все необходимые данные система будет хранить в базе данных MySQL. Для этого в базе данных должны быть таблицы для всех сущностей предметной области и связи между ними.

Серверная часть программы должна быть реализована на языке программирования PHP, интерфейс - на HTML с использованием языка программирования JavaScript, и каскадных таблиц стилей CSS.

Система должна позволять выполнять следующие действия:

1. Заходить в систему под своим логином и паролем. Если введённого сочетания логина и пароля нет в системе, должно выдаваться соответствующее сообщение об ошибке.
2. В систему должна быть встроена система контроля версий, т.е. для каждого файла в базе данных должен храниться набор версий.
3. В системе должны быть формы для редактирования списка программ, со следующими возможностями:
   1. Создания новой программы, для программы задаётся название, автор, дата создания, текст программы.
   2. Удаления существующей программы.
   3. Поиск в базе данных программы по имени, автору и типу станка.
4. Формы для редактирования нового
5. Должна быть возможность привязки к программе документации (чертежей и других файлов). Чертежи загружаются как картинки (фотографии) и показываются на странице программы.
6. РАЗРАБОТКА И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ
7. Установка локального web-сервера denwer

Запустите скачанный инсталлятор Денвера. Вы увидите перед собой нечто вроде следующего:

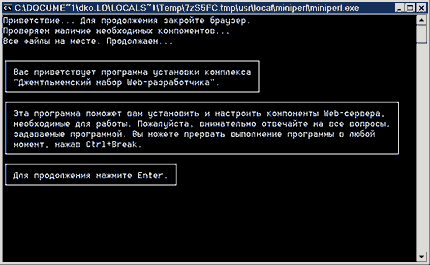


Рис. 9. Установка denwer

Вас спросят о том, в какой каталог вы хотели бы установить комплекс (по умолчанию используется C:\WebServers, вам нужно лишь нажать Enter, чтобы согласиться с этим выбором). В указанном каталоге будут расположены абсолютно все компоненты системы, и вне него никакие файлы в дальнейшем не создаются (исключая ярлыки на Рабочем столе).

Настоятельно рекомендуем вам устанавливать комплекс в каталог первого уровня — то есть, C:\WebServers, а не, например, C:\My\WebServers. Дело в том, что инсталляторы пакетов расширений ищут базовый комплект именно на первом уровне по всем дискам. И, если не находят, заставляют вас ввести имя директории вручную.

Далее вам предложат ввести имя виртуального диска, который будет связан с только что указанной директорией. Рекомендуем вам согласиться со значением по умолчанию (Z:). Важно, что диска с этим именем еще не должно содержаться в системе — чаще всего так и происходит с диском Z:.

После этого начнется копирование файлов дистрибутива, а под конец вам будет задан вопрос, как именно вы собираетесь запускать и останавливать комплекс. Удобнее «Создавать виртуальный диск при загрузке машины».

В некоторых версиях Windows 98 имеется ошибка, в результате которой диск с первого раза не отключается — мои эксперименты и исследования показали, что это именно ошибка в subst, а не что-то иное.

Собственно, вот и все. Вы сможете выполнить установку комплекса, имея всего 2 клавиши на клавиатуре: Enter и Y (чтобы ввести Yes).

После установки на рабочем столе появятся 3 иконки:



Рис. 10. Иконки для работы с denwer.

Для запуска локального web-сервера разработки надо нажать на «Start Denwer».

1. Настройка dbForge Studio для разработки

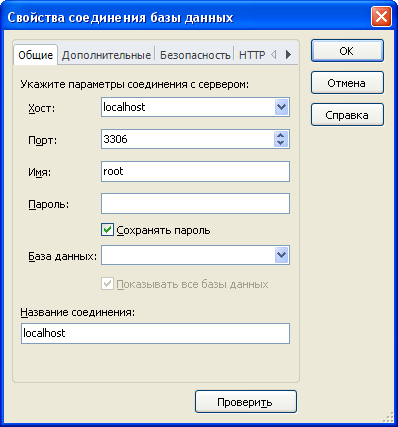


Рис. 11. Настройки подключения к серверу баз данных

**Настройки подключения:**

* Хост: localhost.
* Порт: 3306
* Имя: root.

1. Проектирование Базы Данных

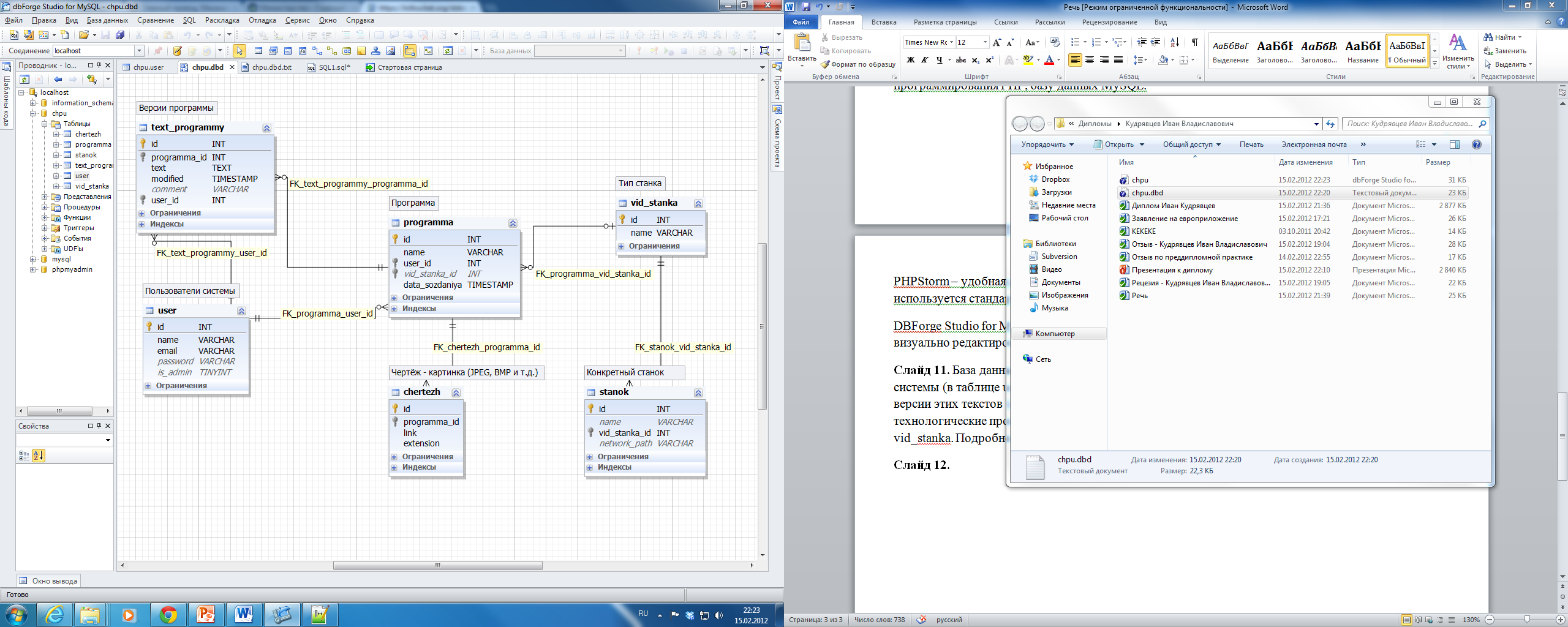
****

Рис. 12. Структура базы данных

**Таблица “user” –** хранение информации о пользователях системы. Нужна для идентификации и авторизации пользователей в системе.

Поля:

* **id –** идентификатор (номер) пользователя в системе (присваивается автоматически);
* **name –** Фамилия имя отчество пользователя (не более 255 символов);
* **email –** электронная почта пользователя, используется как логин (не более 255 символов);
* **password –** md5-хеш сумма пароля пользователя.
* **is\_admin –** Является ли администратором? 0 – не является, 1 – является. Администратор может назначать новых администраторов, а также удалять права администратора у других администраторов.

MD5 – 128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом из Массачусетского технологического института (Massachusetts Institute of Technology, MIT) в 1991 году. Предназначен для создания «отпечатков» или «дайджестов» сообщений произвольной длины и последующей проверки их подлинности.

Хранится вместо самого пароля на случай компрометации базы данных (т.е. когда какой-то кусок базы данных становится доступен злоумышленникам). Если пароль хранится в «прямом» виде, злоумышленники могут незаметно входить под любым пользователем системы.

**Таблица “programma” –** программа для станков с ЧПУ.

Поля:

* **id –** идентификатор (номер) программы в системе (для связей между таблицами);
* **name –** название программы;
* **user\_id –** автор программы;
* **vid\_stanka\_id –** для какого станка предназначена программа;
* **data\_sozdaniya –** дата создания программы, присваивается при .

Для одной программы хранится несколько текстов (версий) программы.

**Таблица “text\_programmy” –** версии текста программы.

Поля:

* **id –** идентификатор (номер) версии программы в системе;
* **programma\_id –** идентификатор программы, к которой относится данный текст (версия);
* **text –** сам текст программы (G-код) или разница (diff) с предыдущей версией программы;
* **modified –** дата и время последней модификации данного текста (данной версии). Для каждой версии своё поле modified.
* **comment –** комментарий к конкретной версии программы, можно хранить любые числа, строки и т.д.
* **user\_id –** пользователь, который создал данную версию программы.

**Таблица “vid\_stanka” –** виды станков.

Поля:

* **id –** идентификатор вида станка;
* **name –** название вида станка (например: Токарный, Зубодолбёжный, Зубофрезерный).

**Таблица “stanok” –** конкретный станок (с номером и т.д.).

Поля:

* **id –** идентификатор станка;
* **name –** идентифицирующее имя станка.
* **vid\_stanka\_id –** вид станка (ссылка на таблицу vid\_stanka).

**Таблица “chertezh” –** чертёж – картинка (JPEG, BMP и т.д.).

Поля:

* **id –** идентификатор станка;
* **programma\_id –** идентификатор программы.
* **link –** имя файла на сервере.
* **extension –** расширение файла.

1. Общая структура проекта

Ниже приведена полная структура директорий, которую мы будем использовать:

/

backend/

common/

components/

config/

params.php

params-local.php \*

lib/

Pear/

yii/

Zend/

migrations/

models/

console/

commands/

SitemapCommand.php

...

config/

main.php

main-local.php \*

params.php

params-local.php \*

runtime/

yiic.php \*

frontend/

components/

config/

main.php

main-local.php \*

params.php

params-local.php \*

controllers/

SiteController.php

...

lib/

models/

ContactForm.php

SearchForm.php

runtime/

views/

layouts/

site/

www/

assets/

css/

js/

index.php \*

yiic

yiic.bat

1. Модели для работы с БД

Для каждой таблицы БД в проекте создана «модель» – класс для удобной работы с этой таблицей, реализующая все необходимые операции.

Модель (model) – это экземпляр класса CModel или класса, унаследованного от него. Модель используется для хранения данных и применимых к ним бизнес-правил.

Модель представляет собой отдельный объект данных. Это может быть запись таблицы базы данных или HTML-форма с полями для ввода данных. Каждое поле объекта данных представляется атрибутом модели. Каждый атрибут имеет метку и может быть проверен на корректность, используя набор правил.

Yii предоставляет два типа моделей: модель формы и Active Record. Оба типа являются расширением базового класса CModel.

Модель формы — это экземпляр класса CFormModel. Она используется для хранения данных, введенных пользователем.Как правило, мы получаем эти данные, обрабатываем, а затем избавляемся от них. Например, на странице авторизации модель такого типа может быть использована для представления информации об имени пользователя и пароле. Подробное описание работы с формами приведено в разделе Работа с формами.

Active Record (AR) — это шаблон проектирования, используемый для абстрагирования доступа к базе данных в объектно-ориентированной форме. Каждый объект AR является экземпляром класса CActiveRecord или класса, унаследованного от него, и представляет отдельную строку в таблице базы данных. Поля этой строки соответствуют свойствам AR-объекта. Подробнее с AR-моделью можно ознакомиться в разделе Active Record.

Модели изначально генерируются при помощи средства Gii (веб-инструмента для генерации кода). Для локального проекта установленного на localhost Gii будет открываться по ссылке: <http://localhost/gii/>

Для защиты требуется ввести Gii-пароль.

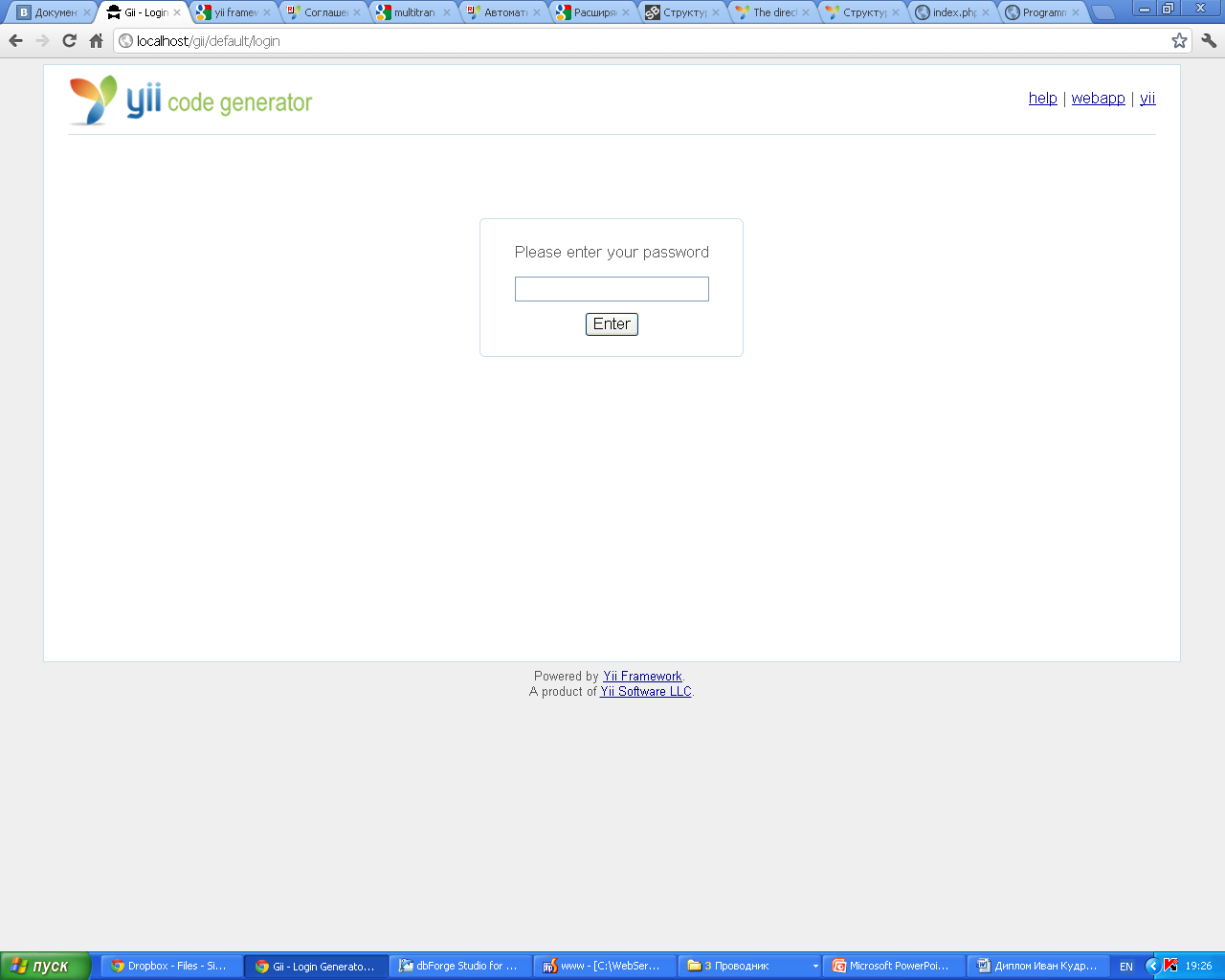


Рис. 13. Вход в Gii.

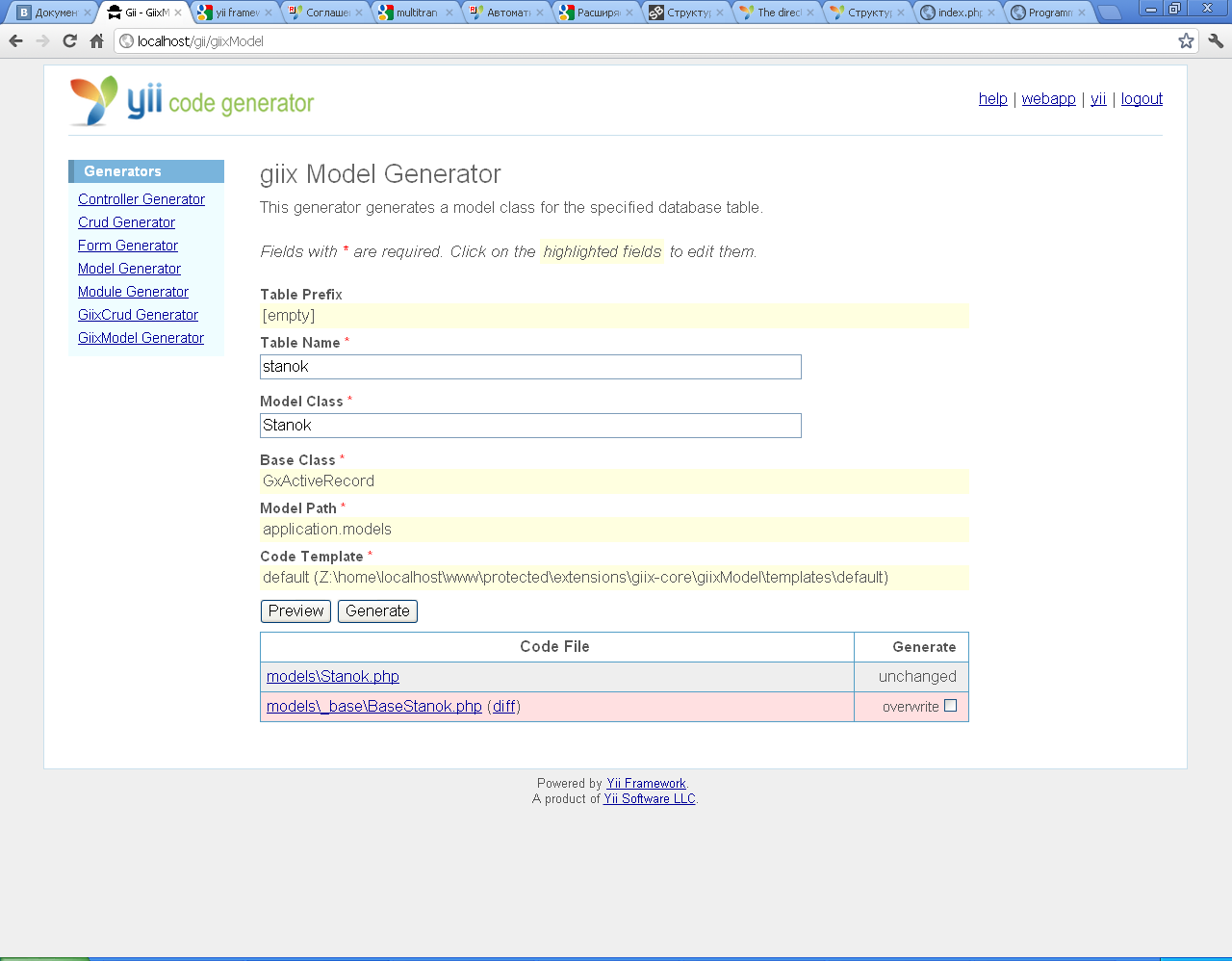


Рис. 14. Генерация шаблона модели в Gii.

Пример сгенерированного при помощи Gii класса:

**<?php**

*/\*\**

\* This is the model base class for the table "programma".

\* DO NOT MODIFY THIS FILE! It is automatically generated by giix.

\* If any changes are necessary, you must set or override the required

\* property or method in class "Programma".

\*

\* Columns in table "programma" available as properties of the model,

\* followed by relations of table "programma" available as properties of the model.

\*

\* ***@property*** *integer $id*

\* ***@property*** *string $name*

\* ***@property*** *integer $user\_id*

\* ***@property*** *integer $vid\_stanka\_id*

\* ***@property*** *string $data\_sozdaniya*

\*

\* ***@property*** *Chertezh[] $chertezhs*

\* ***@property*** *User $user*

\* ***@property*** *VidStanka $vidStanka*

\* ***@property*** *TextProgrammy[] $textProgrammys*

\*/

**abstract class** BaseProgramma **extends** GxActiveRecord

{

**public static function** model($className = \_\_CLASS\_\_)

{

**return** parent::model($className);

}

**public function** tableName()

{

**return 'programma'**;

}

**public static function** label($n = 1)

{

**return** Yii::t(**'app'**, **'Programma|Programmas'**, $n);

}

**public static function** representingColumn()

{

**return 'name'**;

}

**public function** rules()

{

**return array**(

**array**(**'name, user\_id, data\_sozdaniya'**, **'required'**),

**array**(**'user\_id, vid\_stanka\_id'**, **'numerical'**, **'integerOnly'** => true),

**array**(**'name'**, **'length'**, **'max'** => 255),

**array**(**'vid\_stanka\_id'**, **'default'**, **'setOnEmpty'** => true, **'value'** => null),

**array**(**'id, name, user\_id, vid\_stanka\_id, data\_sozdaniya'**, **'safe'**, **'on'** => **'search'**),

);

}

**public function** relations()

{

**return array**(

**'chertezhs'** => **array**(self::HAS\_MANY, **'Chertezh'**, **'programma\_id'**),

**'user'** => **array**(self::BELONGS\_TO, **'User'**, **'user\_id'**),

**'vidStanka'** => **array**(self::BELONGS\_TO, **'VidStanka'**, **'vid\_stanka\_id'**),

**'textProgrammys'** => **array**(self::HAS\_MANY, **'TextProgrammy'**, **'programma\_id'**),

);

}

**public function** pivotModels()

{

**return array**(

);

}

**public function** attributeLabels()

{

**return array**(

**'id'** => Yii::t(**'app'**, **'ID'**),

**'name'** => Yii::t(**'app'**, **'Name'**),

**'user\_id'** => null,

**'vid\_stanka\_id'** => null,

**'data\_sozdaniya'** => Yii::t(**'app'**, **'Data Sozdaniya'**),

**'chertezhs'** => null,

**'user'** => null,

**'vidStanka'** => null,

**'textProgrammys'** => null,

);

}

**public function** search()

{

$criteria = **new** CDbCriteria;

$criteria->compare(**'id'**, $this->id);

$criteria->compare(**'name'**, $this->name, true);

$criteria->compare(**'user\_id'**, $this->user\_id);

$criteria->compare(**'vid\_stanka\_id'**, $this->vid\_stanka\_id);

$criteria->compare(**'data\_sozdaniya'**, $this->data\_sozdaniya, true);

**return new** CActiveDataProvider($this, **array**(

**'criteria'** => $criteria,

));

}

}

1. Система авторизации

Форма входа находится по адресу: <http://localhost/site/login>. **«site»** – это имя Контроллера. **«login»** – это название конкретного действия. Для выполнения авторизации вызывается класс **SiteController** (имя берётся из запроса), а в нём вызывается функция **actionLogin**.

В базе данных хранятся md5-хеши паролей. Для того чтобы заменить пароли на их md5-хеши можно использовать SQL-запрос:

UPDATE user SET password = md5(password)

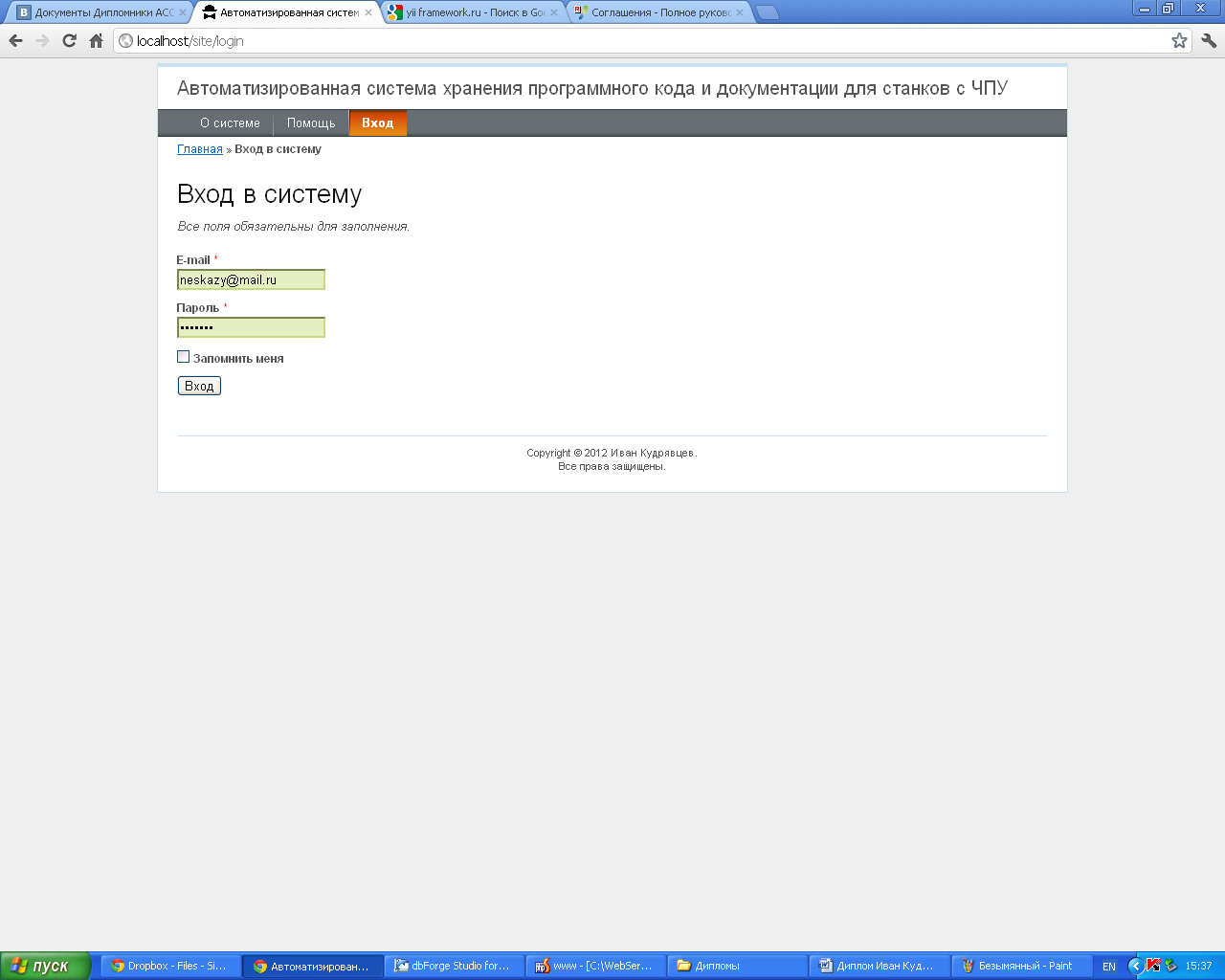
****

Рис. 15. Вход в систему

***/\*\****

**\* Страница входа в систему**

**\*/**

**public function actionLogin()**

**{**

***// Объект - форма входа***

**$model = new LoginForm;**

***// Если это AJAX-проверка во время ввода, то сразу проверяем и отправляем ответ браузеру***

**if (isset($\_POST['ajax']) && $\_POST['ajax'] === 'login-form') {**

**echo CActiveForm::validate($model);**

**Yii::app()->end();**

**}**

***// Если произошла отправка формы***

**if (isset($\_POST['LoginForm'])) {**

***// То сохраняем введённые пользователем данные (логин и пароль)***

**$model->attributes = $\_POST['LoginForm'];**

***// Проверяем их корректность, пробуем войти в систему с введёнными логином и паролем***

**if ($model->validate() && $model->login())**

**$this->redirect(Yii::app()->user->returnUrl); *// Если получилось, то возвращаемся на страницу откуда пришли***

**}**

***// Показываем форму входа пользователю***

**$this->render('login', array('model' => $model));**

**}**

После входа в систему пользователь может менять пароли других пользователей.

1. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ СИСТЕМЫ
2. Вход в систему

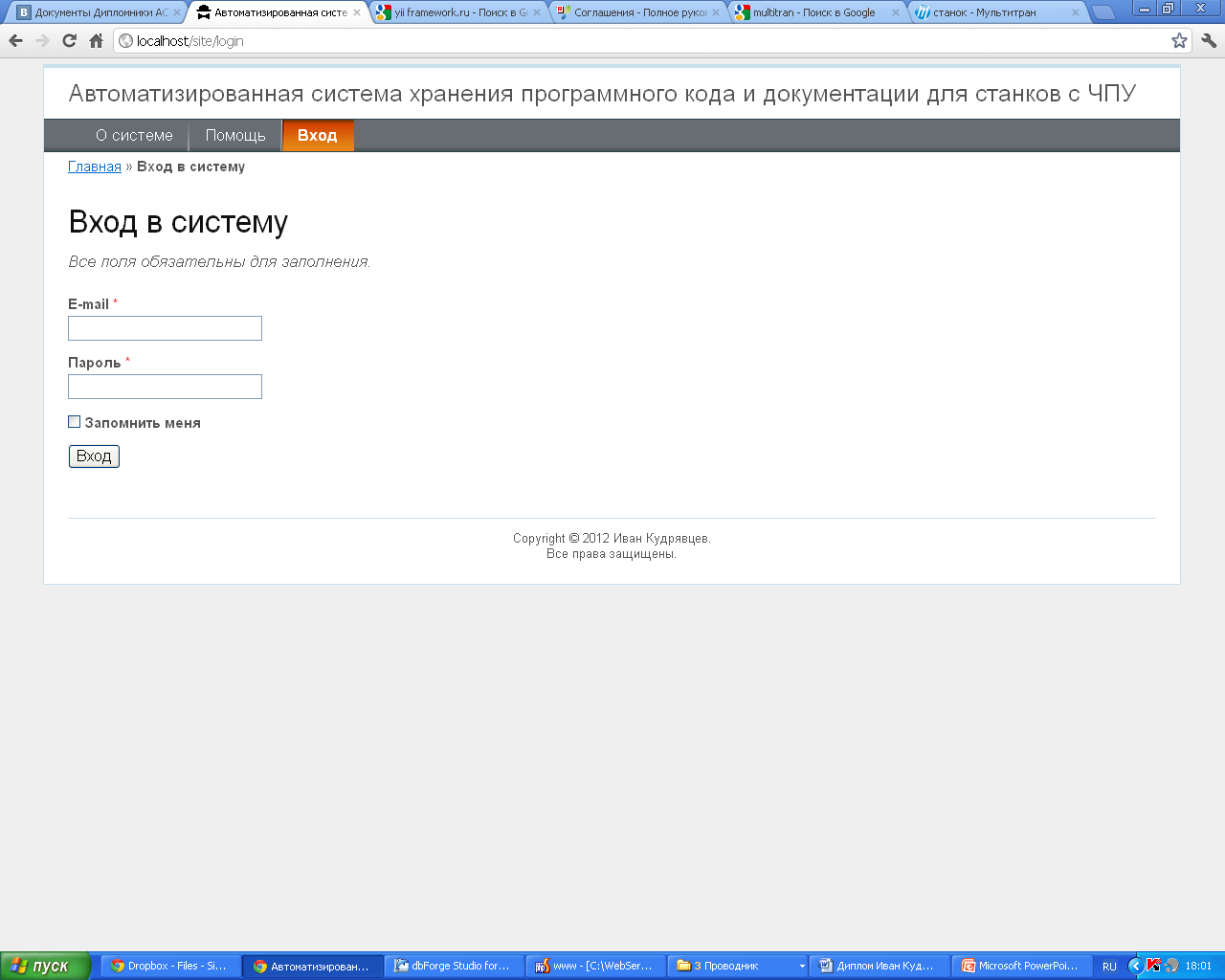


Рис. 16. Окно входа в систему

Для входа в систему нужно ввести свой e-mail и пароль и нажать на кнопку «Вход». Пользователей добавляют администраторы. Администраторы также могут:

* указать для любого пользователя, что он также является администратором;
* снять права администратора с любого пользователя;
* редактировать данные пользователя (фамилию имя отчество и e-mail);
* полностью удалить пользователя из системы;

После входа в систему мы попадаем на страницу «О системе», где рассказывается о достоинствах системы.

Для управления пользователями нужно выбрать пункт меню «Пользователи». При этом появляется список пользователей:

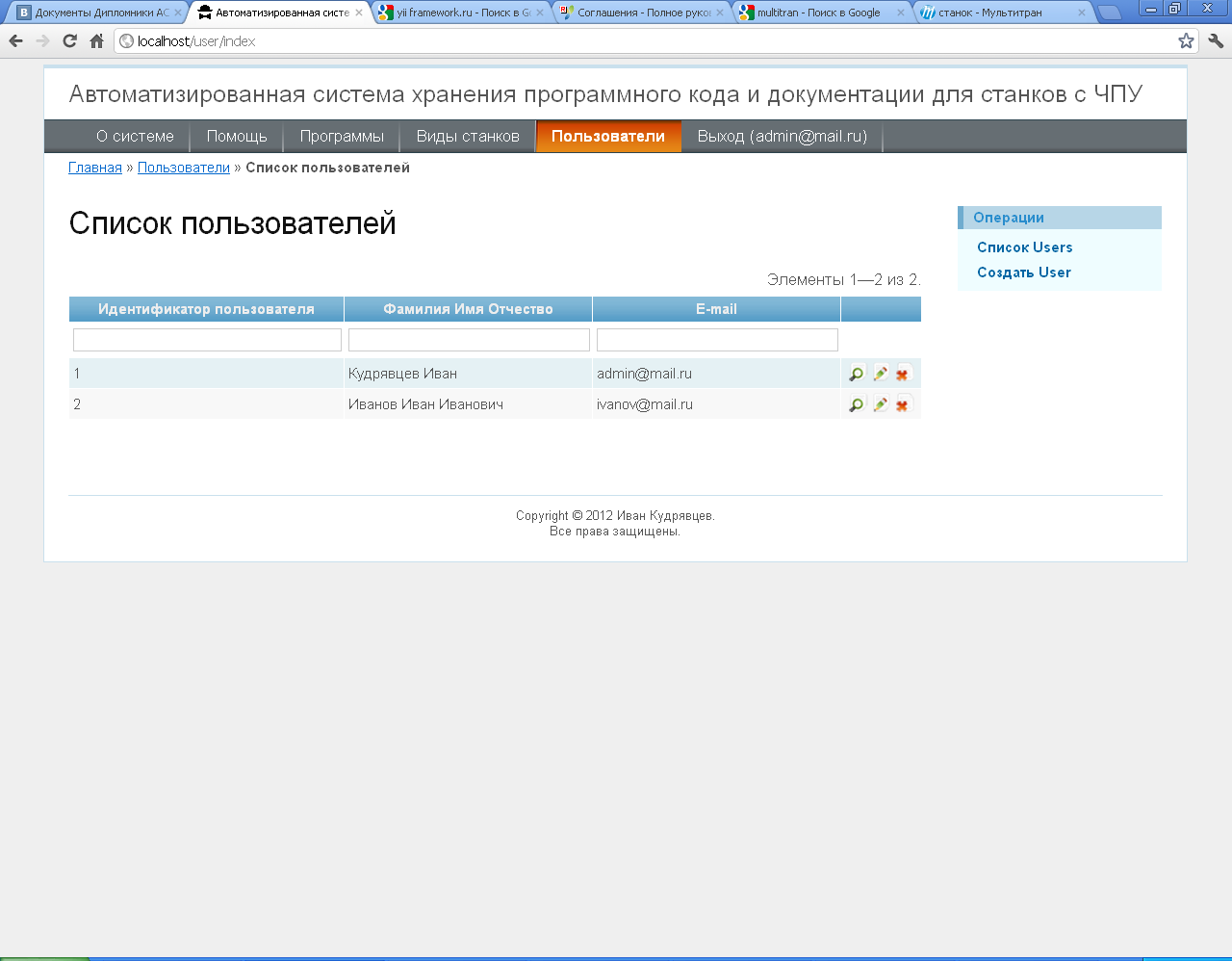


Рис. 17. Список пользователей системы.

В форме «Список пользователей» можно искать пользователей по фамилии, имени и отчеству, а также по любой их части. Также можно искать пользователей по e-mail, редактировать пользователей, просматривать их страницы, на которых перечислены их программы, чертежи и т.д.

Если перейти на страницу конкретного пользователя, то можно посмотреть данные о нём, а также программы, автором которых он является.

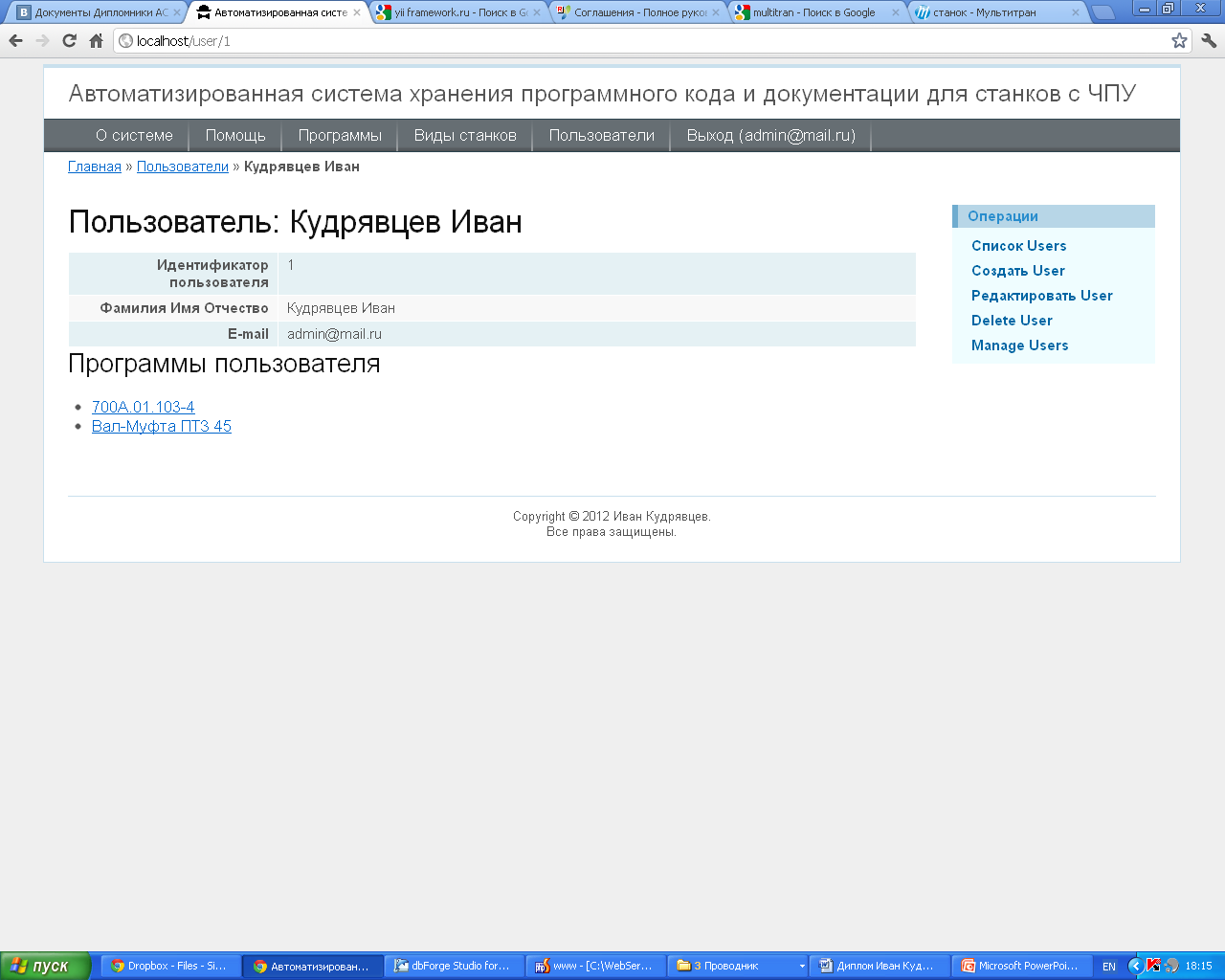


Рис. 18. Страница пользователя системы

Анкету можно отредактировать или удалить.

Если далее перейти по ссылке на программу пользователя.

1. Работа со списком программ

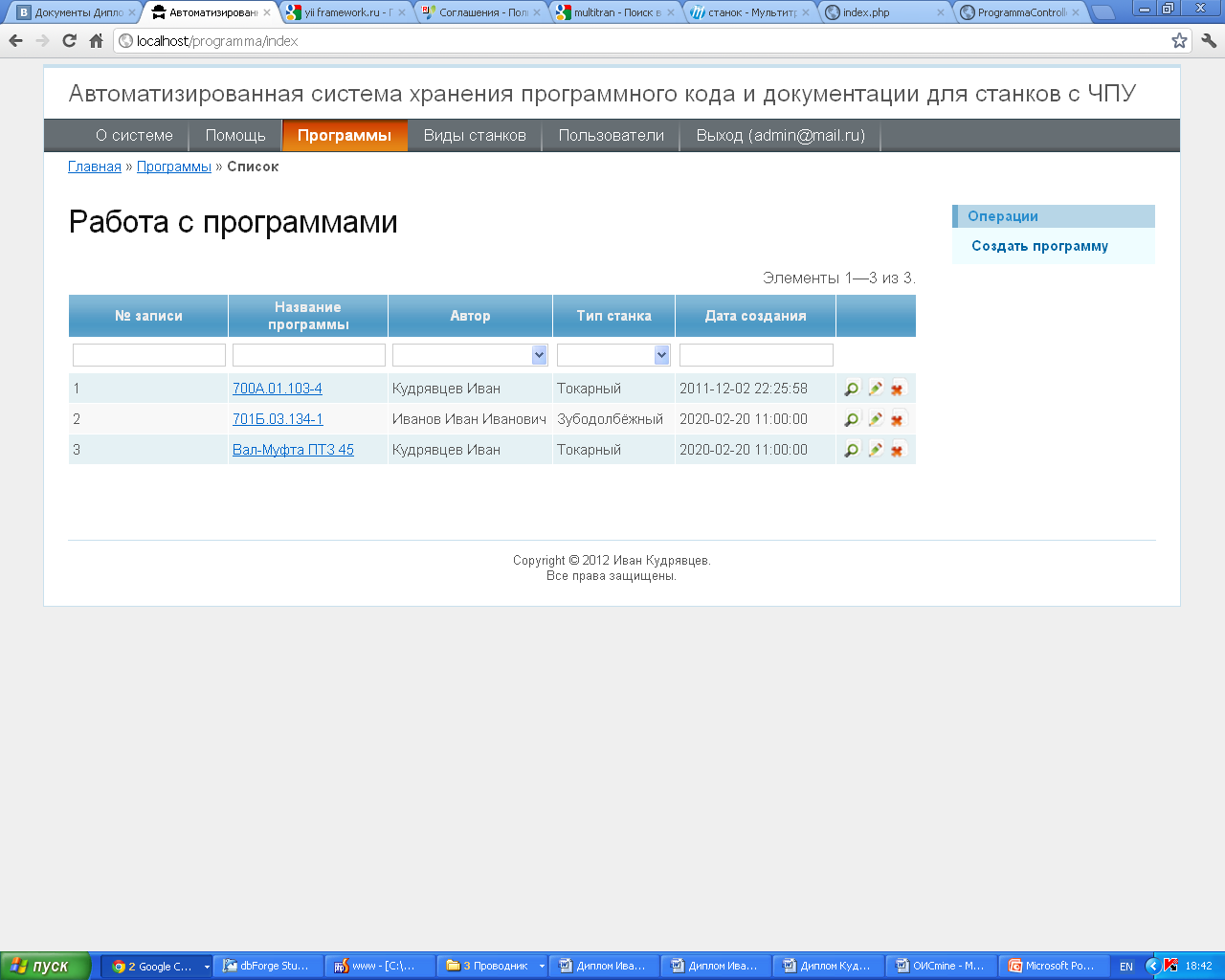


Рис. 19. Работа со списком программ.

В этой форме можно просматривать список программ. Осуществлять поиск по значениям отдельных полей. Например, можно ввести название (или часть названия) программы и искать этому полю. Можно отсортировать программы по названиям, автору, типу станка, дате создания программы. Для сортировки нужно нажать на заголовок таблицы.

При нажатии на название программы происходит переход на страницу этой программы, где указано название, тип станка, автор, дата, а также привязанные к этой программе чертежи и версии программы.

1. Типы станков

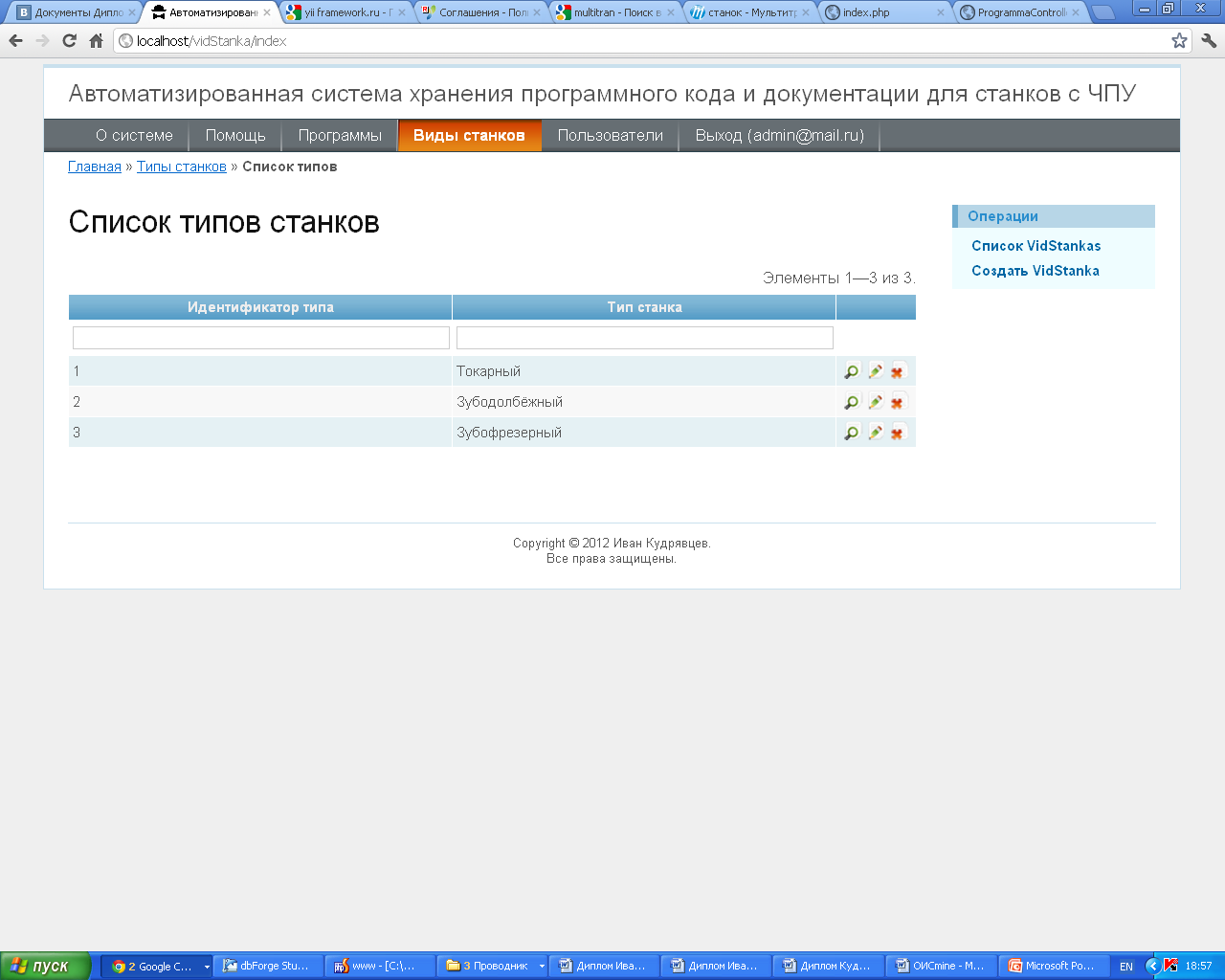


Рис. 20. Список типов станков.

В этой форме можно просматривать, искать и редактировать типы станков. Можно искать по названию типа станка, отредактировать тип станка, удалить тип станка.

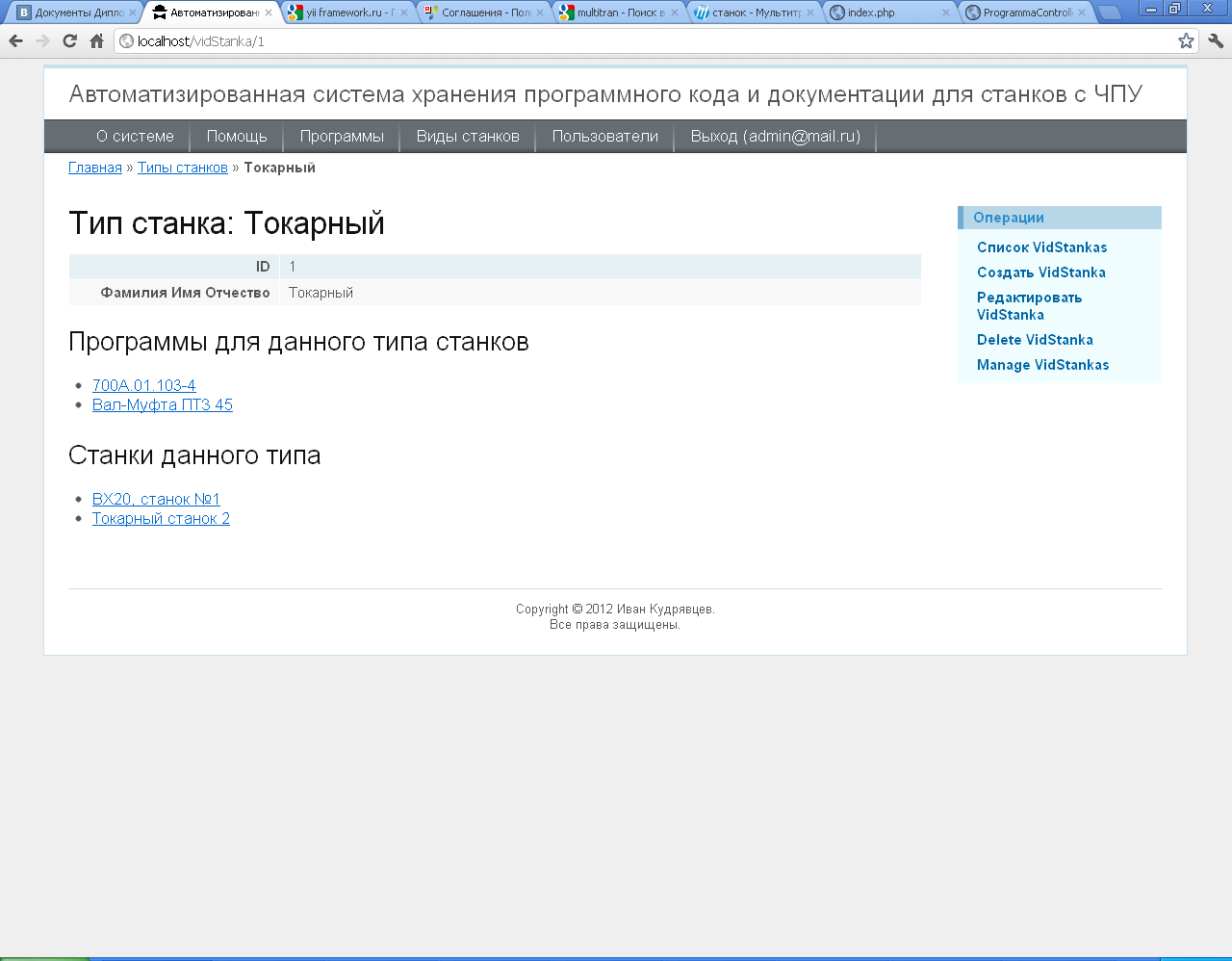


Рис. 21. Просмотр конкретного типа станка.

Внизу страницы показываются программы, которые предназначены для данного типа станка и все станки данного типа. По ссылкам можно перейти на страницы программ и на страницы станков.

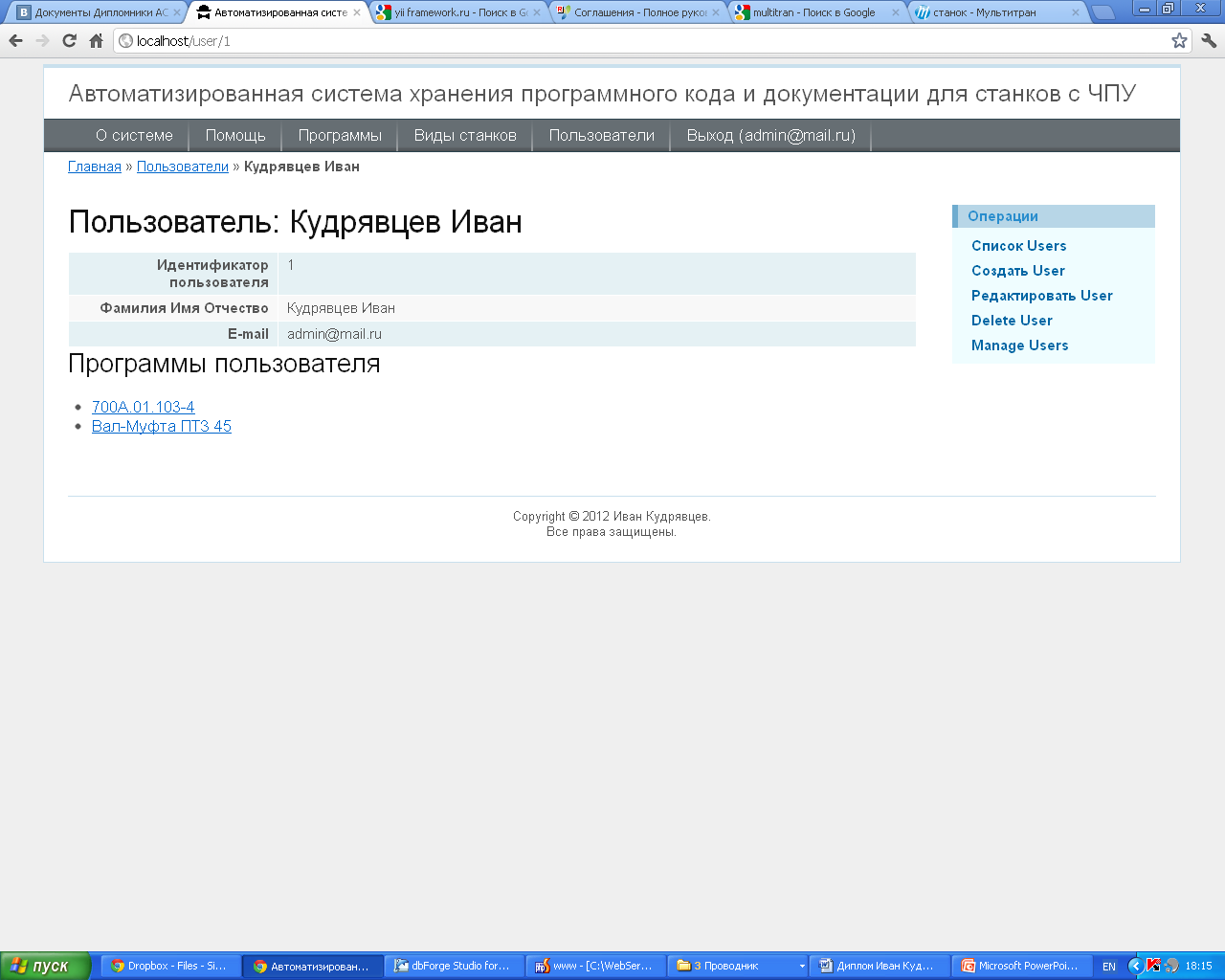


Рис. 22. Страница пользователя системы.

1. Технико-экономическое обоснование
2. Концепция технико-экономического обоснования

Производство – процесс, в котором огромное значение имеет показатель эффективно затраченного времени, т. е. времени, которое ушло непосредственно на изготовление продукции. Чем больше времени уйдет на изготовление продукции, тем выше показатель эффективности производства, а чем выше эффективность производства, тем больше прибыли можно из него выручить. Любое производство представляет собой систему слаженно работающих механизмов, систему взаимодействия людей и машин. Для того, что бы показатель эффективно затраченного времени повышался, необходимо сокращать время, затраченное человеком на наладку системы.

Цель дипломного проекта заключается в разработке автоматизированной системы хранения программного кода и документации для станков с ЧПУ.

Внедрение данной системы даст ряд дополнительных преимуществ, к которым относятся:

* Снижение показателей времени и сил, затраченных на настройку оборудования
* Наличие быстрого доступа к необходимым файлам и сведениям
* Снижению количества ошибок, осуществляемых вследствие неразберихи с документацией и программным кодом
* Укрепление производственной дисциплины

Причинами возникновения задач данной работы послужило отсутствие программных средств, способных обеспечить быструю и качественную наладку оборудования. Отсюда вытекают и цели дипломного проекта: создание автоматизированной системы, с помощью которой стало бы возможно осуществление автоматизации процесса наладки оборудования.

При разработке дипломного проекта были поставлены следующие задачи:

* Разобраться в текущем механизме наладки оборудования
* Выяснить возможность внедрения подобной системы
* Разработать и реализовать базу данных при помощи MySQL
* Разработать и реализовать Web-интерфейс при помощи PHP
* Внедрить разработанную систему в процесс производства на предприятии.

1. Потребительские свойства разрабатываемой научно-технической продукции.

К наиболее значимым потребительским свойствам разрабатываемой, в данном дипломном проекте, научно-технической продукции относятся:

* Наличие Web-интерфейса, обеспечивающего кроссплатформенность и мобильность данного проекта. Это позволяет избежать проблем с совместимостью приложения и платформы, на которой оно будет использоваться. В понятие “мобильность” входит возможность получения доступа к базе данных, содержащей необходимые файлы (программы и документацию) и инструменты для работы с ними из любого места, где есть компьютер и подключение к интернету.
* Объединение в одной базе данных программного кода, чертежей к нему, а так же важной информации, обеспечивающей высокую эффективность процесса наладки оборудования и наглядность действий.
* Обеспечение защиты от несанкционированного доступа к важным файлам, со стороны некомпетентных лиц. Обеспечивается системой авторизации.
* Наличие СКВ (система контроля версий), позволяющей удобно организовывать, отслеживать и редактировать файлы, которые имеют несколько версий.
* Низкие требования к компьютерным ресурсам. Делает систему доступной.
* Простой и понятный интерфейс, что позволяет легко работать с системой даже неопытному пользователю.

При использовании данной автоматизированной системы, потребитель также получает выгоду с экономической точки зрения, увеличивая показатель полезно затраченного на производство времени. Также при использовании данного продукта повышается производственная дисциплинированность, что сокращает возможность ошибок при процессе наладки оборудования.

1. Анализ рынка и план маркетинга

Рынок программного обеспечения, направленный на данную предметную область в России, находится в стадии зарождения. Аналогов данной системы, широко предлагаемых на рынке, обнаружено не было.

Рынок потенциальных потребителей системы состоит из производственных организаций, на которых применяются станки с числовым программным управлением любого типа.

Анализ механизмов наладки станочного оборудования по различным предметным областям, распространяемых на сегодняшний момент в производственных организациях, позволяет говорить о том, что большинство из них не соответствует критериям эффективности, удобства пользования и актуальности.

Период жизненного цикла данного программного обеспечения не ограничен явным образом, в связи с новизной разработанной системы и возможностью совершенствования.

Так как разрабатываемая система имеет специализированную направленность, то рынок потенциальных покупателей будет ограничен производственными организациями, на которых применяются станки с числовым программным управлением любого типа. В этом сегменте рынка определяющими критериями для подобного ПО являются универсальность и эффективность применения, а также соотношение цены и полезного эффекта от использования продукта.

Полезный эффект складывается из следующих показателей: высокая скорость работы системы, минимальное потребление, высокая надежность системы, простота и удобство использования.

Низкие объемы продаж обусловлены спецификой рассматриваемого продукта. Они могут быть увеличены при успешной рекламной политике и адаптации продукта для использования в других системах, где необходимо решать схожие задачи.

1. Производство продукта

Особенностью проектируемой системы является то, что место проектирования и обслуживания не является принципиальным с точки зрения влияния на расходы. Для продвижения программного продукта, требуется лишь наличие персонального компьютера и подключение к интернету. Предполагается, что местом создания и обслуживания является место написания проекта.

Расчет себестоимости выполненных работ осуществляется по следующим статьям:

* расходы на оплату труда (относится заработная плата научных сотрудников, инженеров и прочего инженерно-технического персонала, непосредственно занятых выполнением конкретной работы);
* отчисления на социальные нужды (относят затраты, связанные с выплатой единого социального налога);
* накладные расходы (включаются расходы на управление и хозяйственное обслуживание);
* материалы (относятся затраты на сырье, основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, необходимые для выполнения конкретной работы) [13].

1. Расходы на оплату труда

В работе непосредственно были задействованы только руководитель и исполнитель. Поэтому расходы на выплату заработной платы будут учитываться только в отношении данных лиц.

Сроки проведения работы – 4 месяцев. Ставка руководителя – 30 000 рублей в месяц. Ставка исполнителя – 20 000 рублей в месяц. Представим расчеты в виде таблицы .

Таблица 1 - Расходы на оплату труда.

|  |  |  |
| --- | --- | --- |
| Работник | Величина з/п., руб. | Выплаты по з/п за 4 месяца, руб. |
| Руководитель | 30 000 | 120 000 |
| Исполнитель | 20 000 | 80 000 |
| **Итого** | **50 000** | **200 000** |

Дополнительная заработная плата составляет 12% от суммы основной заработной платы:

С дз = 200 000 \*0,12=24 000 рублей.

1. Отчисления на социальные нужды

Рассчитаем величину отчислений на социальные нужды за четыре месяца.

Расчет производится по формуле

 [1]

где  - величина заработной платы;  - величина дополнительной заработной платы; - Ставка взносов, %. На момент проведения работы ставка единого социального налога равна 26%. Исходя из имеющихся данных, вычислим размер отчислений:

С сн =(200 000 + 24 000)\*0,26 = 58 240 рублей.

1. Расходы на материалы

Стоимость расходных материалов, необходимых для выполнения разработки, определяется исходя из величины их расхода, действующих цен. Расчет расходов по статье "Материалы" представлен в таблице.

Таблица 2 - Расходы на материалы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Единица измерения | Количество | Цена, руб. | Сумма, руб. |
| Бумага для принтера | пачка в 500 листов | 1 | 150 | 150,00 |
| Чернила для принтера | Чернильниц, штук | 4 | 1500 | 6 000,00 |
| Транспортно-заготовительные расходы (10% от материалов) | | | | 615,00 |
| **Итого** | | | | **6 755,00** |

1. Накладные расходы

Накладные расходы составляют 20% от всех расходов .

С нр =(200 000 + 58 240 + 6 755)\*0,2 = 52 999 рублей.

**Калькуляция себестоимости разработки**

Итоговая калькуляция себестоимости приведена в таблице.

Таблица 3 - Калькуляция себестоимости продукции.

|  |  |
| --- | --- |
| **Статья расходов** | **Сумма, руб.** |
| Материалы | 6 755,00 |
| Основная заработная плата | 200 000,00 |
| Дополнительная заработная плата | 24 000 |
| Отчисления на социальные нужды | 58 240 |
| Накладные расходы | 52 999 |
| **ИТОГО** | **341 994** |

В результате произведенных вычислений получаем себестоимость разработки 341 994 рублей.

1. Трудоемкость и календарный план

Календарный план разработки программного продукта можно представить в виде следующей последовательности:

1. Сбор научно-технической документации изучаемой области, изучение существующих методов.
2. Программная реализация Web-интерфейса на языке программирования PHP и реализация базы данных в среде MySQL. Разработка программного продукта.
3. Написание пояснительной записки к дипломному проекту.

Календарный план работ, включающий все вышеописанные этапы сведен в таблице 5.

Таблица 4 Ленточный график проведения разработки программного продукта.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Этапы работы | Исполнитель | Длительность, недели (5 дней) |
| 1 | Руководство проекта, написание технического задания, контроль выполнения этапов проекта и оценка качества полученных результатов. | Руководитель | 16 |
| 2 | Сбор научно-технической документации изучаемой области, изучение существующих методов. | Специалист | 2 |
| 3 | Программная реализация Web-интерфейса на языке программирования PHP и реализация базы данных в среде MySQL. | Специалист | 4 |
| 4 | Разработка программного продукта. | Специалист | 8 |
| 5 | Написание пояснительной записки к дипломному проекту. | Специалист | 2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Этапы и работы** | **Исполнитель** | **Трудоёмкость,**  **чел.-дн.** | **Длительность,**  **дн.** | **Продолжительность работы (пятидневка)**  *Таблица 5* | | | | | | | | | | | | | | | | | | | |
| сен | | | | окт | | | | ноя | | | | дек | | | | янв | | | |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Ознакомление с предметной областью | Руководитель | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 20 | 20 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | Составление задания на дипломное проектирование | Руководитель | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 3 | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | Ознакомление с существующей документацией | Руководитель | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 10 | 10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | Ознакомление с существующей системой на практике | Руководитель | 5 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 15 | 15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | Обработка и анализ информации | Исполнитель | 14 | 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | Проектирование системы | Исполнитель | 35 | 35 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | Реализация | Исполнитель | 14 | 14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 | Отладка и тестирование | Исполнитель | 30 | 30 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Оформление документации | Руководитель | 2 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 7 | 7 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | Сдача проекта | Руководитель | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Исполнитель | 1 | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Комплексная оценка эффективности работ

Научно-технический эффект:

Для определения научно-технического эффекта воспользуемся следующим методом оценки уровня качества НТПР.

Выделим основные параметры (показатели) системы. Оценим их с помощью десятибальной системы. Затем проставим коэффициент значимости этих показателей, который в сумме равен единице.

Последним этапом посчитаем коэффициент качества (Ккач) внедряемой системы, чем больше он единицы, тем выше качество.

*Таблица 6*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Показатель | Бальная оценка | | Коэффициент значимости,α i |
| До внедрения системы, Б базi | После внедрения системы Бi |
| 1 | Скорость поиска информации | 3 | 10 | 0,1 |
| 2 | Вероятность ошибок | 2 | 9 | 0,3 |
| 3 | Трудозатраты на обработку данных | 3 | 9 | 0,3 |
| 4 | Удобство хранения данных | 3 | 9 | 0,1 |
| 5 | Надежность хранения данных | 2 | 10 | 0,3 |

Ккач = ∑ αi × Б i /Б базi,

Ккач = 0,1 × 10/3 + 0,3 × 9/2 + 0,3 × 9/3 + 0,1 × 9/3 + 0,3 × 10/2 = 3,96.

*Вывод:*

В результате расчетов получаем, что коэффициент качества равен 3, 96, что значительно превышает единицу. Это свидетельствует о том, что после внедрения системы значительно повышается качество работы.

**Социальный эффект:**

После внедрения данного проекта, существенно снижается количество времени, которое уходит на наладку оборудования. Также процесс становится более удобным и надежным, что существенно снижает вероятность возникновения ошибок и повышает производственную дисциплину.

**Экономическая эффективность проекта**

Рентабельности инвестиций

, 

где  – чистая прибыль от проекта в году *t*; *T* – количество лет в инвестиционном периоде; *I* – величина инвестиционных затрат.

Период возврата (срок окупаемости) инвестиций равен трем кварталам

Чистая текущая стоимость проекта NPV



NPV = 3239,6 тыс.р.

где - экономия на текущих издержках; - единовременные затраты на проведение научных исследований и совершенствование продукции (технологии);  - экономия на потребности в основных средствах; *R* – ставка дисконтирования, принятая для оценки анализируемого проекта; *T* – время реализации проекта, определяемое как количество интервалов инвестиционного периода, т.е. количество процентных периодов, по окончании которых производится начисление процентов.

Ставка дисконтирования рассчитывается по следующей формуле:

R = Rf + b (Rm – Rf) + g1 + g2 + C = 8,25 + 0,1675\*(15- 8,5) + 1,4 + 1 =11 ,75,

,где Rf – номинальная безрисковая ставка дохода (годовая), b – коэффициент "бета" (мера систематического риска), Rm – среднерыночная ставка дохода, (Rm – Rf) – рыночная премия за риск, g1 – премия для малых предприятий, g2 – премия за риск, характерный для данной компании, C – страновой риск.

Положительное значение *NPV* свидетельствует о целесообразности принятия решения о финансировании и реализации проекта

Показатель внутренней рентабельности проекта

,

где *IRR* – искомая ставка внутренней рентабельности проекта.

*IRR=21%*

Таблица 7 - Оценка экономической эффективности проекта

|  |  |
| --- | --- |
| Параметр | Значение |
| Себестоимость | 341 994 руб. |
| NPV | 3239600 руб. |
| IRR | 21% |
| ROI | 222% |
| Коэффициент качества | 4.2 |

**Вывод:**

Проведя экономическое обоснование научно-технического проекта выяснилось, что разрабатываемая система по уровню качества превосходит уровень существующих технологий наладки примерно в четыре раза. В ходе проведения экономического обоснования были посчитаны необходимые на реализацию проекта затраты. Себестоимость системы составляет 341 994 руб. Также был произведен анализ рынка и разработан план маркетинга для успешной реализации данного проекта. Составлен календарный план. Внедрение данного проекта приводит к сокращению времени простоя оборудования, что прямым образом влияет на увеличение прибыли того или иного предприятия. За счет снижения вероятности возникновения ошибок, снижается количество финансовых ресурсов, затрачиваемых на восстановление оборудования, что тоже является прямой выгодой для предприятия. С экономической точки зрения, внедрение данного проекта является выгодным.

# 5. ОХРАНА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

Вопросы, которые рассматриваются в этом разделе, относятся к вопросам охраны прав интеллектуальной деятельности. Разрабатываемая в дипломном проекте программа «Автоматизированная система хранения документации и программного кода для станков с ЧПУ» является результатом интеллектуальной деятельности.

Программа для ЭВМ

Под *программой для ЭВМ* понимается *"... представленная в объективной форме совокупность данных и команд, предназначенных для функционирования ЭВМ и других компьютерных устройств в целях получения определенного результата"*. Кроме того, в понятие программы для ЭВМ входят *"...подготовительные материалы, полученные в ходе разработки программы для ЭВМ, и порождаемые ею аудиовизуальные отображения"* [12, ст. 1261].

С точки зрения программистов и пользователей программа для ЭВМ представляет собой детализацию алгоритма решения какой-либо задачи и выражена в форме определенной последовательности предписаний, обеспечивающих выполнение компьютером преобразования исходных данных в искомый результат.

Можно выделить следующие объективные формы представления программы для ЭВМ:

* *исходная программа* (или исходный текст) - последовательность предписаний на алгоритмическом (понятном человеку) языке высокого уровня, предназначенных для автоматизированного перевода этих предписаний в последовательность команд в объектном коде;
* *рабочая программа*(или объектный код) - последовательность машинных команд, т. е. команд, представленных на языке, понятном ЭВМ;
* *программа, временно введенная в память ЭВМ* - совокупность физических состояний элементов памяти запоминающего устройства ЭВМ (ОЗУ), сохраняющихся до прекращения подачи электропитания к ЭВМ;
* *программа, постоянно хранимая в памяти ЭВМ* - представленная на языке машины команда (или серия команд), выполненная в виде физических особенностей участка интегральной схемы, сохраняющихся независимо от подачи электропитания.

Исходная и рабочая программы, как правило, представляются в виде записи на том или ином языке, выполненной на бумаге или машиночитаемом носителе данных: магнитном или оптическом диске, магнитной ленте и т. п.

Предоставляемая законодательством правовая охрана распространяется *"... на все виды программ для ЭВМ (в том числе на операционные системы и программные комплексы), которые могут быть выражены на любом языке и в любой форме, включая исходный текст и объектный код …"* [12, ст.1261]. Так как преобразование исходного текста программы для ЭВМ в объектный (машинный) код с помощью специальных программ-трансляторов не меняет сущности данной программы как произведения, то если охраняется исходный текст программы, значит, охране подлежит и соответствующий ей объектный код. Обратное тоже справедливо.

Правовая охрана программ для ЭВМ распространяется только в отношении формы их выражения и *«… не распространяется на идеи, концепции, принципы, методы, процессы, системы, способы, решения технических, организационных или иных задач, открытия, факты, языки программирования»* [12, ст.1259, п. 5].

Авторское право на программу для ЭВМ и базу данных

Предпосылкой охраноспособности программы для ЭВМ и базы данных является их творческий характер, т. е. они должны быть продуктом личного творчества автора. Творческий характер деятельности автора предполагается до тех пор, пока не доказано обратное [12, ст. 1257].

Момент возникновения авторского права является важнейшим юридическим фактом, который устанавливается в силу создания произведения (программы для ЭВМ или базы данных). *"Для возникновения, осуществления и защиты авторских прав не требуется регистрация произведения или соблюдение каких-либо иных формальностей"* [12, ст.1259, п.4].

Часто возникает вопрос: насколько необходимо для возникновения прав на программу для ЭВМ или базу данных их обнародование? Закон устанавливает, что это не является обязательным условием: *“Авторские права распространяется как на обнародованные, так и на необнародованные произведения, выраженные в какой-либо объективной форме …”* [12, ст. 1259, п. 3].

Таким образом, только сам факт создания программы или базы данных, зафиксированных в объективной форме, является основанием возникновения авторского права на эти объекты. С этого момента права автора или иного правообладателя защищаются законом.

Права в отношении программ для ЭВМ и баз данных подразделяются на *личные неимущественные и исключительные права***.**

*Личные права* включают *право авторства, право на имя и право на неприкосновенность* (целостность), *право на обнародование* программы для ЭВМ или базы данных [12, ст. 1265-1268]. Они связаны непосредственно с автором программы для ЭВМ или базы данных: принадлежат лицу, чьим творческим трудом созданы программа для ЭВМ или база данных - автору, являются неотчуждаемыми, т. е. не могут быть переуступлены другому лицу, и не ограничены каким-либо сроком [12, ст. 1228].

*Исключительные права* непосредственно связаны с понятием **"***использования"* программ для ЭВМ и баз данных: *"Автору произведения или иному правообладателю принадлежит исключительное право использовать произведение … в любой форме и любым не противоречащим закону способом …"* [12, ст. 1270, п.1]. При этом под использованием понимается осуществление определенных действий с программами для ЭВМ или базами данных, а именно: опубликование (выпуск в свет); воспроизведение (полное или частичное) в любой форме, любыми способами; распространение; модификацию и иное использование [12, ст. 1270, п.2]. Они могут принадлежать автору или иному правообладателю (гражданину или юридическому лицу). Правообладатель может распоряжаться исключительным правом на произведение [12, ст. 1270, п.1], в том числе по своему усмотрению разрешать или запрещать другим лицам использование [12, ст. 1229, п.1]. Распоряжение принадлежащим правообладателю исключительным правом может осуществляться любым, не противоречащим закону и существу такого исключительного права способом, в том числе путем его отчуждения по договору другому лицу или предоставления другому лицу права использования [12, ст. 1233, п.1]. Срок действия исключительного права ограничен

Каждая составляющая понятия использования программы для ЭВМ или базы данных имеет конкретное содержание, которое также определено законом:

* *воспроизведение* - *"... изготовление одного или более экземпляров произведения или его части в любой материальной форме, … в том числе запись в память ЭВМ"* [12, ст. 1270, п. 2, п.п.1];
* *распространение* – предоставление доступа к произведению *"... путем продажи или иного отчуждения его оригинала или экземпляров"* [12, ст. 1270, п. 2, п.п.2].
* *публичный показ (выпуск в свет***) -** *"... любая демонстрация оригинала или экземпляров произведения непосредственно … … либо с помощью технических средств в месте, открытом для свободного посещения, или в месте, где присутствует значительное число лиц … "* [12, ст. 1270, п. 2, п.п.3];

Обращает на себя внимание то, что понятие "использование" не связано с функционированием программы (или ее выполнением) с целью получения результата. Последнее лучше относить к понятию "потребление" или "пользование" программы. Поэтому каждый раз, когда пользователь запускает программу для того, чтобы произвести расчеты, построить графики или таблицы и т. п., он не "использует" (поскольку не создается новый экземпляр), а "потребляет" программу, не нарушая при этом ничьих прав.

В целях *оповещения о своих правах* правообладатель *"... вправе использовать знак охраны авторского права, который помещается на каждом экземпляре произведения и состоит из следующих элементов: латинской буквы С в окружности; имени или наименования правообладателя; года первого опубликования произведения"* [12, ст. 1271]. Знак охраны авторского права может проставляться на упаковке, на самом программном продукте, а также на всех сопроводительных и дополнительных материалах, однако это не является обязательным. Следует иметь в виду, что сам знак ничего не защищает (защищает закон!), и что наличие или отсутствие знака охраны никак не связано с возникновением авторского права. Знак охраны авторского права - это цивилизованная форма предупреждения желающих использовать данный объект, что права на него охраняются законом, с одновременным сообщением о том, кому эти права принадлежат.

Исключительные права на программу или базу данных переходят по наследству в установленном законом порядке, и их можно реализовать в течение срока действия авторского права.

Права на программу для ЭВМ или базу данных не связаны с *правом собственности на материальный носитель,* на котором они зафиксированы. Передача прав на материальный носитель не влечет за собой передачи каких-либо прав на программу для ЭВМ или базу данных [12, ст.1227]. Иными словами, передача носителя информации (например, дискеты) с зафиксированной на нем программой третьему лицу не означает передачи каких-либо прав на эту программу.

*Правообладание*

Если человек (или группа людей) самостоятельно, по личной инициативе создал программу для ЭВМ или базу данных, то он является одновременно и автором, и правообладателем созданного произведения, что позволяет ему по собственному усмотрению использовать эту программу (или базу данных) в личных целях, продавать, раздавать бесплатно, разрешать тиражировать и распространять или иным образом распоряжаться своими исключительными правами.

Если права на программу для ЭВМ или базу данных принадлежат одновременно нескольким авторам, то их действия по использованию произведения и распоряжению исключительными правами на произведение лучше всего урегулировать в специальном договоре между всеми авторами.

Если программа для ЭВМ или база данных разрабатываются по заказу, то заказчик заключает с разработчиком договор на создание данного объекта. В этом случае, *"… когда программа для ЭВМ или база данных создана по договору, предметом которого было ее создание (по заказу), исключительное право на такую программу или такую базу данных принадлежит заказчику, если договором между подрядчиком (исполнителем) и заказчиком не предусмотрено иное"* [12, ст.1296, п.1]. Если программа или база данных создана по договорам, *"…которые прямо не предусматривали ее создание, исключительное право на такую программу или базу данных принадлежит подрядчику (исполнителю), если договором между ним и заказчиком не предусмотрено иное".* В этом случае заказчик сохраняет за собой право использования программы или базы данных, но возможность распоряжения исключительными правами на такие программы или базы должна быть закреплена в договоре между заказчиком и исполнителем[12, ст.1297, п.1]. В обоих случаях, когда автору созданных по заказу (или иному договору) программы для ЭВМ или базы данных не принадлежит исключительное право, он имеет право на вознаграждение [12, ст. 1296, п.4, ст.1297, п.3].

Если автор (работник) связан с каким-либо предприятием (работодателем) трудовыми отношениями и создал программу или базу данных в пределах установленных для него трудовых обязанностей, то в этом случае исключительное право на созданную программу для ЭВМ или базу данных принадлежит работодателю, если трудовым или иным договором между работодателем и автором не предусмотрено иное, а такая программа или база данных называется служебным произведением [12, ст. 1295, п. 1, 2]. Работодатель в данном случае правомочен распоряжаться программой для ЭВМ или базой данных по своему усмотрению. За автором сохраняются только личные права, и автор не имеет права распоряжаться исключительными правами и предоставлять для использования программу для ЭВМ или базу данных другим физическим или юридическим лицам. Однако, если предварительно между автором и работодателем был заключен договор в отношении использования созданной программы для ЭВМ или базы данных и распоряжения исключительными правами на них, то руководствуются условиями договора.

Кроме личных (неимущественных) прав автору “служебной” программы для ЭВМ (базы данных) принадлежит право на вознаграждение при условии использования работодателем созданных произведений или передачи исключительного права другому лицу. Размер и порядок выплаты этого определяется договором [12, ст. 1295, п. 2,]. К сожалению, такие договоры заключаются редко, и напрасно, так как только при заключении договора автор наиболее эффективно может реализовать свое право на вознаграждение за каждый вид использования созданной им программы для ЭВМ или базы данных. Для работодателя такой договор тоже целесообразен, так как наличие в нем условия о выплате вознаграждения автору позволяет дополнительно стимулировать разработчика к созданию конкурентоспособных программы или базы.

*Передача исключительных прав на программу для ЭВМ и базу данных*

Исключительные права на программу для ЭВМ и базу данных могут быть переданы в полном объеме (отчуждение исключительных прав) другим физическим или юридическим лицам. Передача исключительных прав должна быть оформлена заключением договора в письменной форме, который называется **“***Договор об отчуждении исключительного права***”** [12, ст.1234].

При полной уступке исключительных прав автором или иным правообладателем правоприобретатель (физическое или юридическое лицо) получает возможность осуществлять все эти права в полном объеме: использовать программу для ЭВМ или базу данных самостоятельно, например, путем изготовления и распространения экземпляров программ или баз данных, разрешать и запрещать третьим лицам их использование, передать все права или отказаться от них вовсе, - в то время как прежний правообладатель лишается этой возможности.

Обладатель исключительного права на программу для ЭВМ или базу данных может предоставить другому физическому или юридическому лицу право использования соответствующих объектов по *лицензионному договору* [12, ст.1235, п.1]. В лицензионном договоре должны быть определены следующие основные условия: способы использования объекта, порядок выплаты вознаграждения и срок действия договора, а также территория, на которой используется данная программа для ЭВМ или база данных [12, ст. 1235, п.3-6]. Законом предусмотрены лицензионные договоры двух видов: *исключительная* лицензия и *неисключительная* (простая) лицензия. В случае исключительной лицензии передаются права на монопольное использование объекта лицензии одному лицу без сохранения за лицензиаром права выдачи лицензий другим лицам. В случае неисключительной лицензии передаются права на использование объекта лицензии в пределах, обусловленных договором, при этом правообладатель оставляет такие же права за собой и может выдавать неисключительные лицензии на тот же объект на тех же или иных условиях другим лицам [12, ст.1236].

*Нарушение прав на программу для ЭВМ и базу данных*

Специфика программ для ЭВМ и баз данных такова, что они очень уязвимы в смысле их незаконного использования (прежде всего, путем копирования и распространения копий). Незаконно изготовленные (скопированные) или используемые экземпляры программы для ЭВМ или базы данных называются *контрафактными,* а несанкционированное использование чужих программ или баз данных путем опубликования (выпуска в свет), воспроизведения (полного или частичного), распространения, иного использования считается нарушением исключительных прав на программы для ЭВМ или базы данных, т. е. *нарушением авторского права.*

**ПРОГРАММА ДЛЯ ЭВМ**

**Автоматизированная система хранения документации и программного кода для станков с ЧПУ**

Описание программы

15 листов

Правообладатель: Кудрявцев Иван Владиславович

Автор: Кудрявцев Иван Владиславович

(ф., и., о.)

© Кудрявцев И. В., 2012

Санкт-Петербург

2012

**СОСТАВ ПРОГРАММНОГО КОМПЛЕКСА**

bazachpu.EXE - приложение

**Р Е Ф Е Р А Т**

Автор: Кудрявцев Иван Владиславович

Правообладатель: Кудрявцев Иван Владиславович

Программа для ЭВМ: Автоматизированная система хранения документации и программного кода для станков с ЧПУ

Аннотация: Система предназначена для решения проблемы хранения документации и программного кода для станков с числовым программным управлением. Она обеспечивает выполнение следующих действий: 1) Оперативное проектирование структуры разнообразных анкет и других документов управления кооперацией и форм их представления; 2) Формирование и ведение разнообразных справочников, используемых при заполнении документов; 3) Формирование типизированных атрибутов документов; 4) Сбор и накопление анкетных данных; 5) Обработка анкетных данных в соответствии с запросами специалистов.

Тип ЭВМ: IBM PC/AT и совместимые с ней

ОС: Windows XP/Vista/7

Язык программирования: PHP

**ЛИЦЕНЗИОННЫЙ ДОГОВОР**

**НА ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ ДЛЯ ЭВМ**

**Стороны в Договоре:**

Гражданин Кудрявцев Иван Владиславович, проживающий по адресу: 197101, г. Санкт-Петербург, Большой пр. П. С. д. 71 кв. 14, именуемый в дальнейшем "ЛИЦЕНЗИАР", с одной стороны, и

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Санкт-Петербургский государственный электротехнический университет "ЛЭТИ" им. В.И.Ульянова (Ленина)", именуемый в дальнейшем "ЛИЦЕНЗИАТ", в лице проректора по научной работе деятельности КУТУЗОВА В.М., действующего на основании Доверенности, с другой стороны,

**принимая во внимание:**

1. что Лицензиар является автором и правообладателем программы для ЭВМ "Автоматизированная система хранения документации и программного кода для станков с ЧПУ»;
2. Лицензиат желает получить на условиях настоящего Договора лицензию на использование упомянутой программы для ЭВМ с целью проведения научных исследований в области медицинского приборостроения;
3. Лицензиар готов предоставить Лицензиату такую лицензию,

**договорились о следующем.**

**1. Термины и их определения**

1. "ПРОГРАММА ДЛЯ ЭВМ (ПрЭВМ)" - программное обеспечение " Автоматизированная система хранения документации и программного кода для станков с ЧПУ ".
2. "ДОКУМЕНТАЦИЯ" - комплект документов, передаваемых Лицензиаром Лицензиату, включающий руководство пользователя по применению и обслуживанию программы для ЭВМ.
3. "ПРОИЗВОДСТВЕННАЯ ПЛОЩАДКА" - научные лаборатории и кафедры Лицензиата.
4. "РАБОЧЕЕ МЕСТО" - конкретная ЭВМ, на которой используется Программа для ЭВМ.

**2. Предмет Договора**

1. Лицензиар предоставляет Лицензиату на срок действия настоящего Договора и за вознаграждение, уплачиваемое Лицензиатом, неисключительную лицензию на использование ПрЭВМ. При этом Лицензиату предоставляется право на установку ПрЭВМ не более чем на 10 (десяти) Рабочих местах.
2. Лицензиар передает Лицензиату Документацию к ПрЭВМ.
3. Предоставленное Лицензиату в рамках настоящего Договора право ограничено Производственной площадкой.
4. Лицензиар осуществляет авторский контроль за соблюдением объемов использования ПрЭВМ по настоящему Договору, при этом Лицензиат обеспечивает возможность такого контроля.
5. Лицензиар сохраняет за собой право самому использовать ПрЭВМ и предоставлять неисключительные лицензии на право ее использования третьим лицам.

**3. Обеспечение Договора**

1. Лицензиар передает Лицензиату ПрЭВМ в объеме и виде, достаточном для ее использования, и Документацию в течение 15 (пятнадцати) дней со дня подписания настоящего Договора. ПрЭВМ передается Лицензиату в виде диска в количестве 1 (одной) штуки, содержащей ПрЭВМ. По факту передачи ПрЭВМ и Документации составляется акт сдачи-приемки с перечнем переданных материалов, подписываемый обеими Сторонами.
2. Если Лицензиат установит неполноту или неправильность полученных ПрЭВМ или Документации, то Лицензиар в течение 15 (пятнадцати) дней после сообщения ему об этом Лицензиатом обязан передать недостающие материалы или устранить недостатки ранее переданных ПрЭВМ и Документации.
3. Для оказания помощи в освоении ПрЭВМ Лицензиар по просьбе Лицензиата оказывает консультации пользователям ПрЭВМ.
4. Для целей использования ПрЭВМ в объеме, предусмотренном п. 2.1 настоящего Договора, Лицензиат может изготавливать в необходимом ему количестве копии ПрЭВМ и копии Документации.

**4. Усовершенствования**

1. Лицензиар обязуется незамедлительно информировать Лицензиата о всех произведенных им усовершенствованиях ПрЭВМ и, при желании Лицензиата, передать ему в согласованные сроки новые варианты ПрЭВМ. В отношении новых вариантов ПрЭВМ, переданных Лицензиаром Лицензиату, распространяются все условия настоящего Договора.
2. Лицензиат обязуется предоставлять Лицензиару информацию об использовании ПрЭВМ, которая могла бы быть полезной для усовершенствования ПрЭВМ.

**5. Платежи**

1. За предоставление прав, предусмотренных настоящим Договором, Лицензиат выплачивает Лицензиару единовременное вознаграждение в размере 20000(двадцать тысяч) рублей.
2. Вознаграждение, предусмотренное п. 5.1 настоящего Договора, выплачивается Лицензиатом в течение 30 (тридцати) дней, следующих после подписания акта приемки-сдачи.

**6. Реклама**

1. Лицензиат обязуется при опубликовании результатов исследований, полученных с использованием ПрЭВМ, сообщать в рекламных целях, что исследования производились с использованием ПрЭВМ Лицензиара с указанием авторского права Лицензиара.

**7. Защита передаваемых прав**

1. Лицензиат обязуется не вносить самовольно каких-либо изменений в ПрЭВМ и До
2. кументацию и не дополнять их какими-либо комментариями. Подобные изменения или дополнения возможны только с согласия Лицензиара.
3. Лицензиат обязуется предпринимать все необходимые меры для предотвращения несанкционированного копирования ПрЭВМ и Документации третьими лицами, а также несанкционированной передачи ПрЭВМ и Документации работниками Лицензиата третьим лицам.
4. Если Лицензиату станет известно о противоправном использовании ПрЭВМ третьими лицами, то он незамедлительно сообщит об этом Лицензиару.

**8. Ответственность Сторон и разрешение споров**

1. За невыполнение или ненадлежащее выполнение обязательств по настоящему Договору Стороны несут имущественную ответственность в соответствии с действующим законодательством.
2. Стороны освобождаются от ответственности за неисполнение или ненадлежащее исполнение обязательств, принятых по настоящему Договору, если неисполнение явилось следствием обстоятельств непреодолимой силы (форс-мажор).
3. Сторона, нарушившая свои обязательства по настоящему Договору, освобождается от ответственности за неисполнение или ненадлежащее исполнение этих обязательств, если это нарушение было вызвано причинами, за которые отвечает другая Сторона.
4. В случае возникновения споров между Лицензиаром и Лицензиатом по вопросам, предусмотренным настоящим Договором, Стороны примут все меры к разрешению их путем переговоров между собой. В случае невозможности разрешения указанных споров путем переговоров они будут разрешаться в порядке, предусмотренном действующим законодательством.

**9. Срок действия Договора и условия его расторжения**

1. Настоящий Договор заключен на срок 2 года и вступает в силу с даты его подписания обеими Сторонами.
2. По истечении срока действия настоящего Договора Лицензиат вправе использовать ПрЭВМ, включая усовершенствованные варианты, на Производственной площадке на любом количестве Рабочих мест. При этом обязательства Лицензиата, предусмотренные пп. 7.1 и 7.2 настоящего Договора, сохраняются бессрочно.
3. Действие настоящего Договора по обоюдному согласию Сторон может быть досрочно прекращено, но не ранее чем через три месяца после предложения об этом одной из Сторон. При этом Лицензиат не освобождается от обязательств по платежам, возникшим до расторжения настоящего Договора.
4. Настоящий Договор может быть досрочно расторгнут в одностороннем порядке со стороны Лицензиара из-за невыполнения Лицензиатом своих обязательств по пп. 7.1 или 7.2. В этом случае Лицензиат лишается права дальнейшего использования ПрЭВМ в любой форме и обязан вернуть ее Лицензиару.
5. Если Лицензиат откажется от дальнейшего использования ПрЭВМ, то он уничтожит все имеющиеся у него копии ПрЭВМ.

**10. Заключительные положения**

1. Все изменения и дополнения к настоящему Договору действительны только в тех случаях, если они совершены в письменной форме и подписаны обеими Сторонами.
2. Стороны не имеют права передавать свои права и обязательства по настоящему Договору третьим лицам без письменного согласия на то другой Стороны.
3. Во всем остальном, что не предусмотрено условиями настоящего Договора, будут применяться нормы законодательства Российской Федерации.

**11. Адреса Сторон**

1. ЛИЦЕНЗИАР: Кудрявцев Иван Владиславович, адрес: 197101, Санкт-Петербург, Большой пр. П. С. д. 71 кв. 14.
2. ЛИЦЕНЗИАТ: СПбГЭТУ, адрес: 197376, Санкт-Петербург, ул. Проф. Попова, д. 5.

Настоящий Договор составлен в двух экземплярах для каждой из Сторон и подписан "31"\_\_\_01\_\_\_\_\_ 2011 г. в г. Санкт-Петербурге.

|  |  |
| --- | --- |
| **ЛИЦЕНЗИАР:** | **От ЛИЦЕНЗИАТА:** |
| \_\_\_\_\_\_\_\_\_\_\_ И. В. Кудрявцев | Проректор по научной работе СПбГЭТУ  \_\_\_\_\_\_\_\_\_\_\_\_ В.М. КУТУЗОВ |

ПРИЛОЖЕНИЕ

Листинг модуля **protected/views/programma/index**:

**<?php**

**$this->breadcrumbs = array(**

**'Программы' => array('index'),**

**'Список',**

**);**

**$this->menu = array(**

**array('label' => 'Создать программу', 'url' => array('create')),**

**);**

**?>**

**<h1>Работа с программами</h1>**

**<?php $this->widget('zii.widgets.grid.CGridView', array(**

**'id' => 'programma-grid',**

**'dataProvider' => $model->search(),**

**'filter' => $model,**

**'columns' => array(**

**'id',**

**array(**

**'name' => 'name',**

**'type' => 'raw',**

**'value' => '$data->link()',**

**),**

**array(**

**'name' => 'user\_id',**

**'value' => 'GxHtml::valueEx($data->user)',**

**'filter' =>**

**GxHtml::listDataEx(**

**User::model()->findAllAttributes(null, true)),**

**),**

**array(**

**'name' => 'vid\_stanka\_id',**

**'value' => 'GxHtml::valueEx($data->vidStanka)',**

**'filter' =>**

**GxHtml::listDataEx(**

**VidStanka::model()->findAllAttributes(null, true)),**

**),**

**'data\_sozdaniya',**

**array(**

**'class' => 'CButtonColumn',**

**),**

**),**

**)); ?>**

Листинг модуля **protected/controllers/ProgrammaController.php**

**<?php**

***/\*\****

**\* Действия с программой**

**\*/**

**class ProgrammaController extends GxController**

**{**

***/\*\****

**\* Просмотр программы**

**\* *@param $id Идентификатор программы***

**\*/**

**public function actionView($id)**

**{**

**$this->render('view', array(**

**'model' => $this->loadModel($id, 'Programma'),**

**));**

**}**

***/\*\****

**\* Создание новой программы**

**\*/**

**public function actionCreate()**

**{**

**$model = new Programma;**

**if (isset($\_POST['Programma'])) {**

**$model->setAttributes($\_POST['Programma']);**

**if ($model->save()) {**

**if (Yii::app()->getRequest()->getIsAjaxRequest())**

**Yii::app()->end();**

**else**

**$this->redirect(array('view', 'id' => $model->id));**

**}**

**}**

**$this->render('create', array('model' => $model));**

**}**

***/\*\****

**\* Редактирование программы**

**\* *@param $id Идентификатор программы***

**\*/**

**public function actionUpdate($id)**

**{**

**$model = $this->loadModel($id, 'Programma');**

**if (isset($\_POST['Programma'])) {**

**$model->setAttributes($\_POST['Programma']);**

**if ($model->save()) {**

**$this->redirect(array('view', 'id' => $model->id));**

**}**

**}**

**$this->render('update', array(**

**'model' => $model,**

**));**

**}**

***/\*\****

**\* Удаление программы**

**\* *@param $id Идентификатор программы***

**\* *@throws CHttpException Если действие вызвано не через Ajax-запрос***

**\*/**

**public function actionDelete($id)**

**{**

**if (Yii::app()->getRequest()->getIsPostRequest()) {**

**$this->loadModel($id, 'Programma')->delete();**

**if (!Yii::app()->getRequest()->getIsAjaxRequest())**

**$this->redirect(array('admin'));**

**} else**

**throw new CHttpException(400, Yii::t('app', 'Your request is invalid.'));**

**}**

***/\*\****

**\* Начальная страница: таблица со списком программ, фильтрами и т.д.**

**\*/**

**public function actionIndex()**

**{**

**$model = new Programma('search');**

**$model->unsetAttributes();**

**if (isset($\_GET['Programma']))**

**$model->setAttributes($\_GET['Programma']);**

**$this->render('index', array(**

**'model' => $model,**

**));**

**}**

**}**

Листинг модуля **protected/components/UserIndentity.php**:

**<?php**

***/\*\****

**\* UserIdentity хранит данные необходимые для индентификации и авторизации пользователя**

**\* и содержит метод authenticate, который проверяет правильность логина и пароля.**

**\*/**

**class UserIdentity extends CUserIdentity**

**{**

***/\*\****

**\* Аутентификация пользователя**

**\* *@return boolean true если аутентификация успешна (пользователь залогинен) и false в противном случае.***

**\*/**

**public function authenticate()**

**{**

***// Ищем пользователя по введённому e-mail***

**$user = User::model()->findByAttributes( *// Ищем по атрибутам***

**array(**

**'email' => $this->username *// $this->username - то что ввёл пользователь***

**)**

**);**

***/\*\* @var $user User \*/***

**if (*is\_null*($user)) *// Пользователь не найден***

**$this->errorCode = self::ERROR\_USERNAME\_INVALID;**

**else if ($user->password !== *md5*($this->password)) *// Пароль неверный***

**$this->errorCode = self::ERROR\_PASSWORD\_INVALID;**

**else**

**$this->errorCode = self::ERROR\_NONE; *// Иначе всё "в порядке"***

**return !$this->errorCode;**

**}**

**}**

Листинг формы для редактирования текста программы:

**<div class="form">**

**<?php $form = $this->beginWidget('GxActiveForm', array(**

**'id' => 'programma-form',**

**'enableAjaxValidation' => false,**

**));**

**?>**

**<p class="note">**

**<?php echo Yii::t('app', 'Fields with'); ?> <span**

**class="required">\*</span> <?php echo Yii::t('app', 'are required'); ?>.**

**</p>**

**<?php echo $form->errorSummary($model); ?>**

**<div class="row">**

**<?php echo $form->labelEx($model, 'name'); ?>**

**<?php echo $form->textField($model, 'name', array('maxlength' => 255)); ?>**

**<?php echo $form->error($model, 'name'); ?>**

**</div>**

***<!-- row -->***

**<div class="row">**

**<?php echo $form->labelEx($model, 'user\_id'); ?>**

**<?php echo $form->dropDownList($model, 'user\_id', GxHtml::listDataEx(User::model()->findAllAttributes(null, true))); ?>**

**<?php echo $form->error($model, 'user\_id'); ?>**

**</div>**

***<!-- row -->***

**<div class="row">**

**<?php echo $form->labelEx($model, 'vid\_stanka\_id'); ?>**

**<?php echo $form->dropDownList($model, 'vid\_stanka\_id', GxHtml::listDataEx(VidStanka::model()->findAllAttributes(null, true))); ?>**

**<?php echo $form->error($model, 'vid\_stanka\_id'); ?>**

**</div>**

***<!-- row -->***

**<div class="row">**

**<?php echo $form->labelEx($model, 'data\_sozdaniya'); ?>**

**<?php echo $form->textField($model, 'data\_sozdaniya'); ?>**

**<?php echo $form->error($model, 'data\_sozdaniya'); ?>**

**</div>**

***<!-- row -->***

**<div class="row">**

**<?php echo $form->labelEx($model, 'text\_programmy'); ?>**

**<?php echo $form->textField($model, 'text\_programmy'); ?>**

**<?php echo $form->error($model, 'text\_programmy'); ?>**

**</div>**

***<!-- row -->***

**<label><?php echo GxHtml::encode($model->getRelationLabel('chertezhs')); ?></label>**

**<?php echo $form->checkBoxList($model, 'chertezhs', GxHtml::encodeEx(GxHtml::listDataEx(Chertezh::model()->findAllAttributes(null, true)), false, true)); ?>**

**<label><?php echo GxHtml::encode($model->getRelationLabel('textProgrammys')); ?></label>**

**<?php echo $form->checkBoxList($model, 'textProgrammys', GxHtml::encodeEx(GxHtml::listDataEx(TextProgrammy::model()->findAllAttributes(null, true)), false, true)); ?>**

**<?php**

**echo GxHtml::submitButton(Yii::t('app', 'Save'));**

**$this->endWidget();**

**?>**

**</div>*<!-- form -->***

Листинг программы **protected/controllers/** **SiteController.php**:

**<?php**

***/\*\****

**\* Общие действия сайта**

**\*/**

**class SiteController extends Controller**

**{**

***/\*\****

**\* Специальные действия, вынесенные в отдельные классы.**

**\*/**

**public function actions()**

**{**

**return array(**

***// Отображение картинки CAPTCHA для защиты от ботов.***

**'captcha' => array(**

**'class' => 'CCaptchaAction',**

**'backColor' => 0xFFFFFF,**

**),**

***// Отображение статических страниц из каталога 'protected/views/site/pages'***

***// Доступны по URL: index.php?r=site/page&view=FileName***

**'page' => array(**

**'class' => 'CViewAction',**

**),**

**);**

**}**

***/\*\****

**\* Страница приветствия системы (описание системы).**

**\*/**

**public function actionIndex()**

**{**

**$this->render('index');**

**}**

***/\*\****

**\* Отображение сообщений об ошибках в системе.**

**\*/**

**public function actionError()**

**{**

**if ($error = Yii::app()->errorHandler->error) {**

**if (Yii::app()->request->isAjaxRequest)**

**echo $error['message'];**

**else**

**$this->render('error', $error);**

**}**

**}**

***/\*\****

**\* Показываем страницу контактов**

**\*/**

**public function actionContact()**

**{**

**$model = new ContactForm;**

**if (isset($\_POST['ContactForm'])) {**

**$model->attributes = $\_POST['ContactForm'];**

**if ($model->validate()) {**

**$headers = "From: {$model->email}\r\nReply-To: {$model->email}";**

***mail*(Yii::app()->params['adminEmail'], $model->subject, $model->body, $headers);**

**Yii::app()->user->setFlash('contact', 'Thank you for contacting us. We will respond to you as soon as possible.');**

**$this->refresh();**

**}**

**}**

**$this->render('contact', array('model' => $model));**

**}**

***/\*\****

**\* Страница входа в систему**

**\*/**

**public function actionLogin()**

**{**

***// Объект - форма входа***

**$model = new LoginForm;**

***// Если это AJAX-проверка во время ввода, то сразу проверяем и отправляем ответ браузеру***

**if (isset($\_POST['ajax']) && $\_POST['ajax'] === 'login-form') {**

**echo CActiveForm::validate($model);**

**Yii::app()->end();**

**}**

***// Если произошла отправка формы***

**if (isset($\_POST['LoginForm'])) {**

***// То сохраняем введённые пользователем данные (логин и пароль)***

**$model->attributes = $\_POST['LoginForm'];**

***// Проверяем их корректность, пробуем войти в систему с введёнными логином и паролем***

**if ($model->validate() && $model->login())**

**$this->redirect(Yii::app()->user->returnUrl); *// Если получилось, то возвращаемся на страницу откуда пришли***

**}**

***// Показываем форму входа пользователю***

**$this->render('login', array('model' => $model));**

**}**

***/\*\****

**\* Выход пользователя из системы.**

**\*/**

**public function actionLogout()**

**{**

**Yii::app()->user->logout();**

***// Переадресуем снова на страницу входа***

**$this->redirect('login');**

**}**

**}**

Скрипт для создания базы данных MySQL:

**--**

**-- Отключение внешних ключей**

**--**

***/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/*;**

***--***

**-- Установка кодировки, с использованием которой клиент будет посылать запросы на сервер**

**--**

**SET NAMES 'utf8';**

***--***

**-- Установка базы данных по умолчанию**

**--**

**USE chpu;**

***--***

**-- Описание для таблицы chertezh**

**--**

**DROP TABLE IF EXISTS chertezh;**

**CREATE TABLE chertezh (**

**id INT(11) NOT NULL AUTO\_INCREMENT,**

**programma\_id INT(11) NOT NULL,**

**link VARCHAR(255) NOT NULL COMMENT 'Имя файла',**

**extension VARCHAR(255) NOT NULL COMMENT 'Расширение файла',**

**PRIMARY KEY (id),**

**INDEX FK\_chertezh\_programma\_id (programma\_id),**

**CONSTRAINT FK\_chertezh\_programma\_id FOREIGN KEY (programma\_id)**

**REFERENCES programma(id) ON DELETE RESTRICT ON UPDATE RESTRICT**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 1**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Чертёж - картинка (JPEG, BMP и т.д.)';**

***--***

**-- Описание для таблицы programma**

**--**

**DROP TABLE IF EXISTS programma;**

**CREATE TABLE programma (**

**id INT(11) NOT NULL AUTO\_INCREMENT COMMENT 'Идентификатор программы',**

**name VARCHAR(255) NOT NULL COMMENT 'Название программы',**

**user\_id INT(11) NOT NULL COMMENT 'Автор программы',**

**vid\_stanka\_id INT(11) DEFAULT NULL COMMENT 'Для какого станка предназначена',**

**data\_sozdaniya TIMESTAMP DEFAULT CURRENT\_TIMESTAMP COMMENT 'Дата и время добавления',**

**PRIMARY KEY (id),**

**INDEX FK\_programma\_user\_id (user\_id),**

**INDEX FK\_programma\_vid\_stanka\_id (vid\_stanka\_id),**

**CONSTRAINT FK\_programma\_user\_id FOREIGN KEY (user\_id)**

**REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT,**

**CONSTRAINT FK\_programma\_vid\_stanka\_id FOREIGN KEY (vid\_stanka\_id)**

**REFERENCES vid\_stanka(id) ON DELETE RESTRICT ON UPDATE RESTRICT**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 2**

**AVG\_ROW\_LENGTH = 16384**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Программа';**

***--***

**-- Описание для таблицы stanok**

**--**

**DROP TABLE IF EXISTS stanok;**

**CREATE TABLE stanok (**

**id INT(11) NOT NULL AUTO\_INCREMENT,**

**name VARCHAR(255) DEFAULT NULL COMMENT 'Идентифицирующее имя станка',**

**vid\_stanka\_id INT(11) NOT NULL COMMENT 'Вид станка',**

**PRIMARY KEY (id),**

**INDEX FK\_stanok\_vid\_stanka\_id (vid\_stanka\_id),**

**CONSTRAINT FK\_stanok\_vid\_stanka\_id FOREIGN KEY (vid\_stanka\_id)**

**REFERENCES vid\_stanka(id) ON DELETE RESTRICT ON UPDATE RESTRICT**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 3**

**AVG\_ROW\_LENGTH = 8192**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Вид станка';**

***--***

**-- Описание для таблицы text\_programmy**

**--**

**DROP TABLE IF EXISTS text\_programmy;**

**CREATE TABLE text\_programmy (**

**id INT(11) NOT NULL AUTO\_INCREMENT COMMENT 'Идентификатор версии',**

**programma\_id INT(11) NOT NULL COMMENT 'Программа',**

**`text` TEXT NOT NULL COMMENT 'Текст программы',**

**modified TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,**

**`comment` VARCHAR(255) DEFAULT NULL COMMENT 'Комментарий к изменениям',**

**user\_id INT(11) NOT NULL COMMENT 'Пользователь',**

**PRIMARY KEY (id),**

**INDEX FK\_text\_programmy\_programma\_id (programma\_id),**

**INDEX FK\_text\_programmy\_user\_id (user\_id),**

**CONSTRAINT FK\_text\_programmy\_programma\_id FOREIGN KEY (programma\_id)**

**REFERENCES programma(id) ON DELETE RESTRICT ON UPDATE RESTRICT,**

**CONSTRAINT FK\_text\_programmy\_user\_id FOREIGN KEY (user\_id)**

**REFERENCES user(id) ON DELETE RESTRICT ON UPDATE RESTRICT**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 2**

**AVG\_ROW\_LENGTH = 16384**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Версии программы';**

***--***

**-- Описание для таблицы user**

**--**

**DROP TABLE IF EXISTS user;**

**CREATE TABLE user (**

**id INT(11) NOT NULL AUTO\_INCREMENT,**

**name VARCHAR(255) NOT NULL DEFAULT '' COMMENT 'Фамилия Имя Отчество',**

**email VARCHAR(255) NOT NULL COMMENT 'e-mail',**

**`password` VARCHAR(32) DEFAULT NULL COMMENT 'md5-хеш пароля',**

**PRIMARY KEY (id)**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 3**

**AVG\_ROW\_LENGTH = 8192**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Пользователи системы';**

***--***

**-- Описание для таблицы vid\_stanka**

**--**

**DROP TABLE IF EXISTS vid\_stanka;**

**CREATE TABLE vid\_stanka (**

**id INT(11) NOT NULL AUTO\_INCREMENT COMMENT 'Идентификатор вида станка',**

**name VARCHAR(255) NOT NULL COMMENT 'Название вида',**

**PRIMARY KEY (id)**

**)**

**ENGINE = INNODB**

**AUTO\_INCREMENT = 4**

**AVG\_ROW\_LENGTH = 5461**

**CHARACTER SET utf8**

**COLLATE utf8\_general\_ci**

**COMMENT = 'Вид станка';**

***--***

**-- Вывод данных для таблицы chertezh**

**--**

**-- Таблица chpu.chertezh не содержит данных**

**--**

**-- Вывод данных для таблицы programma**

**--**

**INSERT INTO programma VALUES**

**(1, '700A.01.103-4', 1, 1, '2011-12-02 22:25:58');**

***--***

**-- Вывод данных для таблицы stanok**

**--**

**INSERT INTO stanok VALUES**

**(1, 'Токарный станок 1', 1),**

**(2, 'Токарный станок 2', 1);**

***--***

**-- Вывод данных для таблицы text\_programmy**

**--**

**INSERT INTO text\_programmy VALUES**

**(1, 1, 'Program Code \r\nHello world! \r\n', '2011-12-02 22:51:06', NULL, 1);**

***--***

**-- Вывод данных для таблицы user**

**--**

**INSERT INTO user VALUES**

**(1, 'Кудрявцев Иван', 'mymail@mail.ru', '123'),**

**(2, 'Иванов Иван Иванович', 'ivanov@mail.ru', 'ivanov');**

***--***

**-- Вывод данных для таблицы vid\_stanka**

**--**

**INSERT INTO vid\_stanka VALUES**

**(1, 'Токарный '),**

**(2, 'Зубодолбёжный'),**

**(3, 'Зубофрезерный');**

***--***

**-- Включение внешних ключей**

**--**

***/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/*;**

Выводы

* Разработана База Данных и web-интерфейс для работы с ней
* Объединение в одной базе данных программного кода, чертежей к нему, а так же важной информации, обеспечивает высокую эффективность процесса наладки оборудования и наглядность действий.
* Обеспечена защита от несанкционированного доступа к важным файлам.
* Реализована СКВ (система контроля версий), позволяющей удобно организовывать, отслеживать и редактировать файлы, которые имеют несколько версий.
* Снижение вероятности возникновения ошибок в процессе наладки и снижение связанного с ним риска порчи оборудования, а также причинения вреда здоровью.
* Систему можно дорабатывать и внедрять на реальном производстве

СПИСОК ЛИТЕРАТУРЫ

* + 1. Технико-экономическое обоснование дипломных проектов. Под ред. Проф. Беклкшова В.К., М. Высшая школа, 1999. – 65 с.
    2. С.В. Маклаков. CASE – средства разработки информационных систем.- М.: Диалог-Мифи, 2000. – 120 c.
    3. С.А. Орлов. Технологии разработки программного обеспечения. СПб.: Питер, 2002. – 463 c.
    4. Д. Н. Колисниченко - Самоучитель PHP 5.
    5. <http://www.php.su/> - PHP, MySQL и другие веб-технологии.
    6. <http://www.php.ru/> - форум PHP программистов.