


Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Кафедра «Компьютерные технологии»
Специальность «Вычислительная техника и программное
обеспечение»


Допущен к защите
Зав. кафедрой КТ

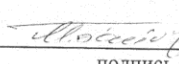

« 16 » _____ 2014 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

Тема «Исследование технологий разработки систем
удаленного доступа»

Магистрант  Коспанов С. М.
подпись (Ф.И.О.)

Руководитель диссертации  Турганбаев Е.С.
подпись (Ф.И.О.)

Рецензент  Байбатшаев М.Ш.
подпись (Ф.И.О.)

Алматы, 2014 г.

**Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»**

Факультет «Информационные технологии»
Специальность «Вычислительная техника и программное обеспечение»
Кафедра «Компьютерные технологии»

ЗАДАНИЕ

на выполнение магистерской диссертации

Магистранту Коспанову Сергею Мухтаровичу
(фамилия, имя, отчество)

Тема диссертации «Исследование технологий разработки систем удаленного доступа»

утверждена Ученым советом университета № ____ от « ____ » _____

Срок сдачи законченной диссертации « ____ » _____

Цель исследования – проектирование и создание лаборатории удаленного доступа для образовательного учреждения.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:

- обоснование актуальности исследования технологий удаленного доступа и их применения в образовании, в частности лабораторий удаленного доступа;
- анализ существующих лабораторий удаленного доступа;
- анализ возможных методов реализации ЛУД и необходимых для этого технологий;
- проектирование и создание ЛУД для образовательного учреждения.

Перечень графического материала (с точным указанием обязательных чертежей):

- обзорные изображения существующих систем;
- структурные схемы систем и использованных технологий, отображающие принципы их работы;
- блок схемы виртуальных приборов.

Рекомендуемая основная литература

- Borisov A.A., Popov N.V., and Shauerman A.A. Foundations of making virtual laboratories in engineering education. 2006, pp. 180–181, 2006.
- Abul K.M., Michael E., V.J. Harward. Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines, 2011.

Г Р А Ф И К
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Теоретическая работа		
1.1 Обоснование выбора темы диссертации	30.11.2012 г.	
1.2 Планирование диссертации	22.12.2012 г.	
1.3 Изучение рынка специализированного программного обеспечения.	30.03.2013 г.	
1.4 Сбор данных о текущем положении в сфере технологий удаленного доступа	30.05.2013 г.	
1.5 Постановка задачи разработки	30.05.2013 г.	
1.6 Анализ эффективности различных возможных подходов к решению задачи	30.06.2013 г.	
1.7 Создание моделей программного обеспечения.	30.09.2013 г.	
1.8. Подготовка статьи в сборник трудов магистрантов	16.11.2013 г.	
1.10. Анализ результатов	01.03.2014 г.	
2. Экспериментальная работа		
2.1. Создание пробной версии системы удаленного доступа	30.11.2013 г.	
2.2. Тестирование функционала системы и улучшение на базе тестирования	30.12.2013 г.	
2.3. Анализ эффективности готовой версии системы	30.01.2014 г.	
2.4 Тестирование в реалистичных условиях(в многопользовательском режиме), исправление если необходимо	28.02.2014 г.	
3. Оформление диссертации	30.04.2014 г.	

Дата выдачи задания _____

Заведующий кафедрой _____

(подпись)

Куралбаев З.К.
(Ф.И.О.)

Руководитель диссертации _____

(подпись)

Турганбаев Е.С.
(Ф.И.О.)

Задание принял к исполнению магистрант _____

(подпись)

Коспанов С.М.
(Ф.И.О.)

Аннотация

Магистерская диссертация посвящена изучению современных технологий удаленного доступа и способов их применения, в частности в образовании. Более глубоко рассмотрены лаборатории удаленного доступа, как наиболее актуальный способ применения систем удаленного доступа в образовании и науке.

Также рассмотрен процесс разработки лаборатории удаленного доступа для образовательного учреждения, включая вопросы выбора технологии создания программного обеспечения, в частности языка программирования и платформ, проектирования и реализации данной системы.

Abstract

This master's thesis is devoted to the study of modern remote access technologies and methods of their use, particularly in education. More deeply examined remote laboratories as the most important way of use remote access systems in education and science.

Also it describes the process of developing remote laboratories for educational institutions; in particular it gives a short review of technologies for creating such software, such as programming languages and platforms, design and implementation methods.

Аңдатпа

Магистрлік диссертация білім жүйесінде қашықтықтан қосылатын заманауи технологияны қолдану тәсілдерін зерттеуге бағытталған. Білім және ғылымда кездесетін қашықтықтан қосылу жүйесін қарастыру үшін, қашықтықтан қосылу зертханалары терең зерттелді. Сонымен қатар білім мекемесінде қашықтықтан қосылу зертханасының құрастырылу процесі де қарастырылған. Мұның құрамына қандай технологияны қолдану керектігі, бағдарламалық жасақтаманы құрастыруды, оның ішінде программалау тілін және платформасының таңдалуы, жобалау мен жүйенің іске асырылуы толығымен қарастырылған.

Содержание

Введение.....	9
1 Аналитическая часть.....	10
1.1 Лаборатории удаленного доступа и их актуальность	10
1.2 Разработка лабораторий удаленного доступа	12
1.2.1 Анализ возможных подходов к разработке ЛУД	12
1.2.2 Примеры существующих лабораторий удаленного доступа	15
2 Теоретическая часть.....	20
2.1 Дистанционное обучение	20
2.2 Задачи	21
2.3 Разработка архитектуры системы	21
2.4 Разработка виртуальной лаборатории	22
2.4.1 Выбор платформы для виртуальной лаборатории	22
2.4.2 Платформа LabVIEW.....	24
2.4.3 Технология Remote Panel	26
2.4.4 LabVIEW Runtime Engine.....	27
2.4.5 Обзор виртуальных приборов.....	27
2.5 Разработка web приложения	30
2.5.1 Выбор платформы для web приложения	30
2.5.2 Язык Ruby	30
2.5.3 Платформа Rails	33
2.5.4 Система управления контентом Comfortable Mexican Sofa.....	35
2.5.5 Система авторизации Devise.....	36
2.5.6 ZURB Foundation framework	37
2.5.7 СУБД PostgreSQL	39
3 Практическая часть	42
3.1 Разработка структуры базы данных	42
3.2 Настройка сервера виртуальных приборов LabVIEW	43
Заключение	46
Список использованных источников	47
Перечень сокращений.....	48
Приложение А	49
Представления приборов:.....	49
Макет приложения:	52
Файлы конфигурации приложения:	53

Введение

Системы удаленного доступа широко применяются в современном мире, практически во всех областях человеческой деятельности.

Технологии удаленной работы на ПК уже давно отработаны и являются надежными и безопасными. Они используются во всем мире крупными корпорациями, предприятиями малого и среднего бизнеса. Данную практику можно так же применять в обучении.

Обучение с элементами удаленного доступа получает все большее распространение. Это обусловлено рядом факторов, в том числе, индивидуальная работа с обучающимися, работа со студентами-инвалидами, работа из дома на рабочем компьютере, а также возможность проведения конференций. Сегодня обучение с элементами удаленного доступа предоставляет гораздо больше возможностей по сравнению с другими формами обучения.

При этом если подача теоретического материала не представляет из себя ничего сложного, так как не подразумевают взаимодействия с обучающимися, удаленно организовать практическую часть обучения зачастую не так легко. В этом помогает концепция лабораторий удаленного доступа.

1 Аналитическая часть

1.1 Лаборатории удаленного доступа и их актуальность

Лаборатория удаленного доступа это набор аппаратных и программных средств, предоставляющих возможность проводить эксперименты на имеющемся оборудовании удаленно.

Принцип, заложенный в основу концепции лабораторий удаленного доступа, уже давно используется в различных областях человеческой деятельности, в особенности в науке и технике. Например, приборы и аппараты, предназначенные для изучения таких объектов, прямой контакт человека с которыми по ряду причин невозможен, всегда управлялись человеком на расстоянии, в том числе и задолго до появления персональных компьютеров и компьютерных сетей. Появление и развитие сети Интернет, приведшее к значительному упрощению электронной связи и давшее возможность легко подключиться с любого персонального компьютера к другому персональному компьютеру или высокопроизводительному серверу в любой точке планеты, позволило сформировать и воплотить в жизнь концепцию удаленного управления оборудованием реальных лабораторий [2]. Помимо этого, как продукт синтеза данного направления с компьютерным моделированием, появились виртуальные лаборатории удаленного доступа, предоставляющие возможность удаленно проводить виртуальные эксперименты.

На сегодняшний день оба типа удаленных лабораторий широко применяются в науке и образовании во многих развитых странах планеты, например: виртуальные лаборатории MIT (<http://ilab.mit.edu/wiki>), сеть удаленных лабораторий России (<http://nano-network.ru/>). Реальные лаборатории пользуются большим интересом в научных областях, так как позволяют исследовать явления и процессы без упрощений присутствующих в моделировании, в то время как виртуальные лаборатории незаменимы в образовании, ввиду большей степени доступности и меньших затратах на содержание.

Преимущества удаленных лабораторий в инженерном образовании следующие [1]:

- Нестрогие временные рамки, позволяют студентам выбирать наиболее удобное время для работы.
- Нестрогие географические рамки, доступны из любого места, обеспеченного доступом к сети.

- Экономия, требует меньше ресурсов, чем обычные лаборатории (меньше площади в здании, меньшие требования по охране труда, меньше косвенных расходов).

- Улучшенное качество эксперимента, ввиду того что его можно повторить в любое время.

- Большая эффективность, так как студенты могут максимально эффективно тратить время на локальных занятиях, репетируя эксперименты вне оных.

- Повышенная безопасность, ввиду отсутствия физического контакта с оборудованием исключена возможность фатальных последствий эксперимента.

Недостатки:

- Недостаточность практики по разрешению проблем с реальным оборудованием.

- Недостаточность практики начальной установки и настройки оборудования.

Тем не менее, создание и внедрение виртуальных лабораторий значительно повышает качество образования.

1.2 Разработка лабораторий удаленного доступа

1.2.1 Анализ возможных подходов к разработке ЛУД

В дистанционной форме обучения построение лабораторных практикумов принципиально отличается от традиционных. Студент должен иметь лабораторию в домашних условиях. Одним из новых направлений является создание автоматизированных виртуальных лабораторий с удалённым доступом [4].

Необходимость создания таких лабораторий обусловлена тем, что инженерное образование предполагает подготовку специалистов-практиков, имеющих навыки работы с приборами, а также для экспериментального закрепления пройденного материала. Лаборатории с удалённым доступом призваны не только дублировать лабораторный практикум очного обучения, но и позволить работать с уникальным дорогостоящим оборудованием, ставить реальные эксперименты из любой точки земного шара. Также может быть реализована возможность работы нескольких студентов за одним лабораторным стендом одновременно [4].

Лаборатория с удаленным доступом подразумевает наличие клиентской части и серверной, однако существует несколько способов организовать обмен данными между пользователями и лабораторным оборудованием:

1. Прямое соединение между внешним сервером и системой лабораторных исследований. Структурная схема такой лаборатории показана на рисунке 1.1:

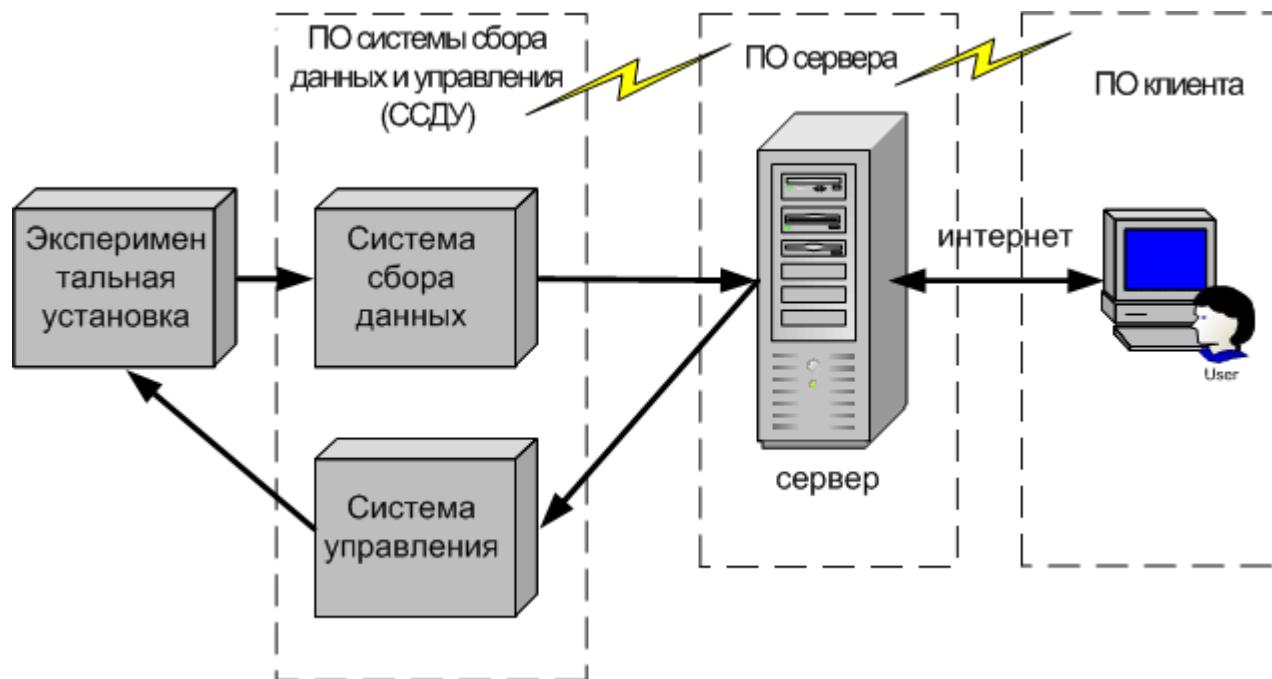


Рисунок 1.1 – Прямое соединение между внешним сервером и системой лабораторных исследований

ПО Системы сбора данных и управления может как являться интерфейсом между экспериментальной установкой так и содержать виртуальную экспериментальную установку в себе; внешний сервер отвечает за обеспечение связи между ПО клиента и ССДУ; ПО клиента обеспечивает удобный пользовательский интерфейс для пользователей.

В таком случае внешний сервер может принимать множество подключений пользователей, однако, следует реализовать возможность совместной работы многих пользователей с одной экспериментальной установкой.

Это можно сделать следующими способами:

а) Сервер принимает все подключения пользователей на прослушиваемый порт, создает новый экземпляр ССДУ с другим сетевым портом прослушки и переключает соединение пользователя на созданный ССДУ.

Таким образом, для каждого соединения затрачивается дополнительная оперативная память сервера в размере экземпляра ССДУ, т.е. затраты ресурсов сервера будут расти пропорционально каждому использованию прибора каждым пользователем. При такой архитектуре запросы пользователей выполняются параллельно.

б) Сервер принимает и обрабатывает все подключения пользователей на 1 порт, используя stateless протокол который не требует поддержания соединения (например http), организуя очередь, и поочередно передает данные от пользователей на единственный экземпляр ССДУ, возвращая результат пользователям.

В данном случае затраты ресурсов будут также расти пропорционально количеству приборов, однако при росте количества пользователей они будут увеличиваться незначительно – только на обработку большей очереди.

Однако быстродействие системы будет резко зависеть от количества пользователей, использующих систему одновременно, так как запросы пользователей будут выполняться последовательно, в порядке очереди.

2. Соединение внешнего сервера с приборами через сервер приложений:

Структурная схема показана на рисунке 1.2.

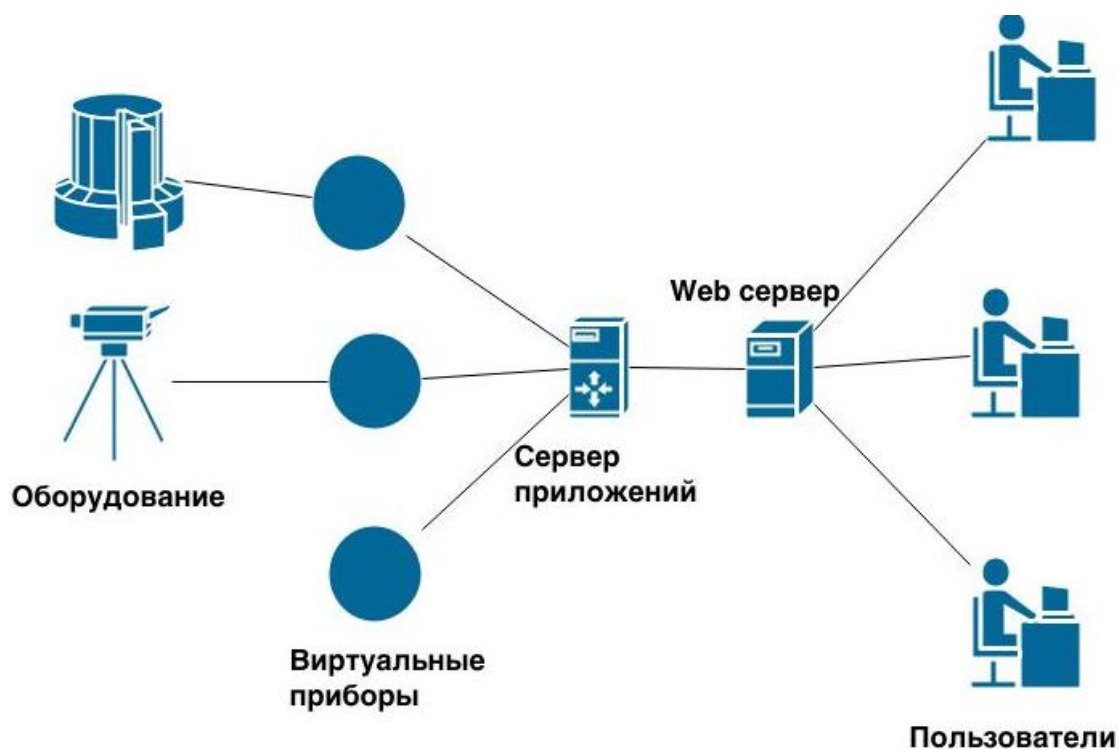


Рисунок 1.2 – Структурная схема удаленной лаборатории, вариант 2

На данной структурной схеме WEB сервер обеспечивает связь между ПО пользователей и сервером приложений, в то время как сервер приложений отвечает за организацию одновременного доступа нескольких пользователей к одному виртуальному прибору. Виртуальные приборы выполняют функции ССДУ и могут быть как интерфейсами между реальными приборами и сервером приложений, так и самостоятельными виртуальными экспериментальными установками.

Таким образом внешний сервер принимает подключение пользователей и перенаправляет их на сервер приложений. Сервер приложений передает данные на нужный виртуальный прибор, который может как выполнять роль ССДУ для реального прибора, так и функционировать самостоятельно. Виртуальный прибор возвращает обработанные данные серверу приложений, сервер приложений передает данные на внешний сервер, откуда они поступают на компьютеры пользователей.

При такой архитектуре затраты ресурсов сервера аналогичны предыдущему случаю (рисунок 3), а также запросы пользователей выполняются параллельно, и при этом не требуется создавать отдельный экземпляр ССДУ для каждого пользователя. Минусом здесь является то что при такой архитектуре невозможно создать длительное соединение между компьютером пользователя и виртуальным прибором, что, при удаленном исследовании длительных процессов может вызвать дополнительные сложности.

1.2.2 Примеры существующих лабораторий удаленного доступа

1.2.2.1 NetLab

NetLab (<http://netlab.unisa.edu.au/index.xhtml>) – Австралийская лаборатория удаленного доступа, предназначенная для использования как преподавательским составом для обучения и демонстраций во время лекций, так и студентами для проведения их экспериментов удаленно на реальном лабораторном оборудовании.

Приложение предоставляет пользователям графические интерфейсы пользователя (GUIs), которые похожи на фактические лабораторные инструменты. Они щелкают кнопками и поворачивают кнопки с их мышью, взаимодействуя с ней, как они были бы с реальным устройством.

NetLab предоставляет удаленный доступ к реальному оборудованию, а не виртуальным моделирующим приборам.

Архитектура NetLab представлена на рисунке 1.3

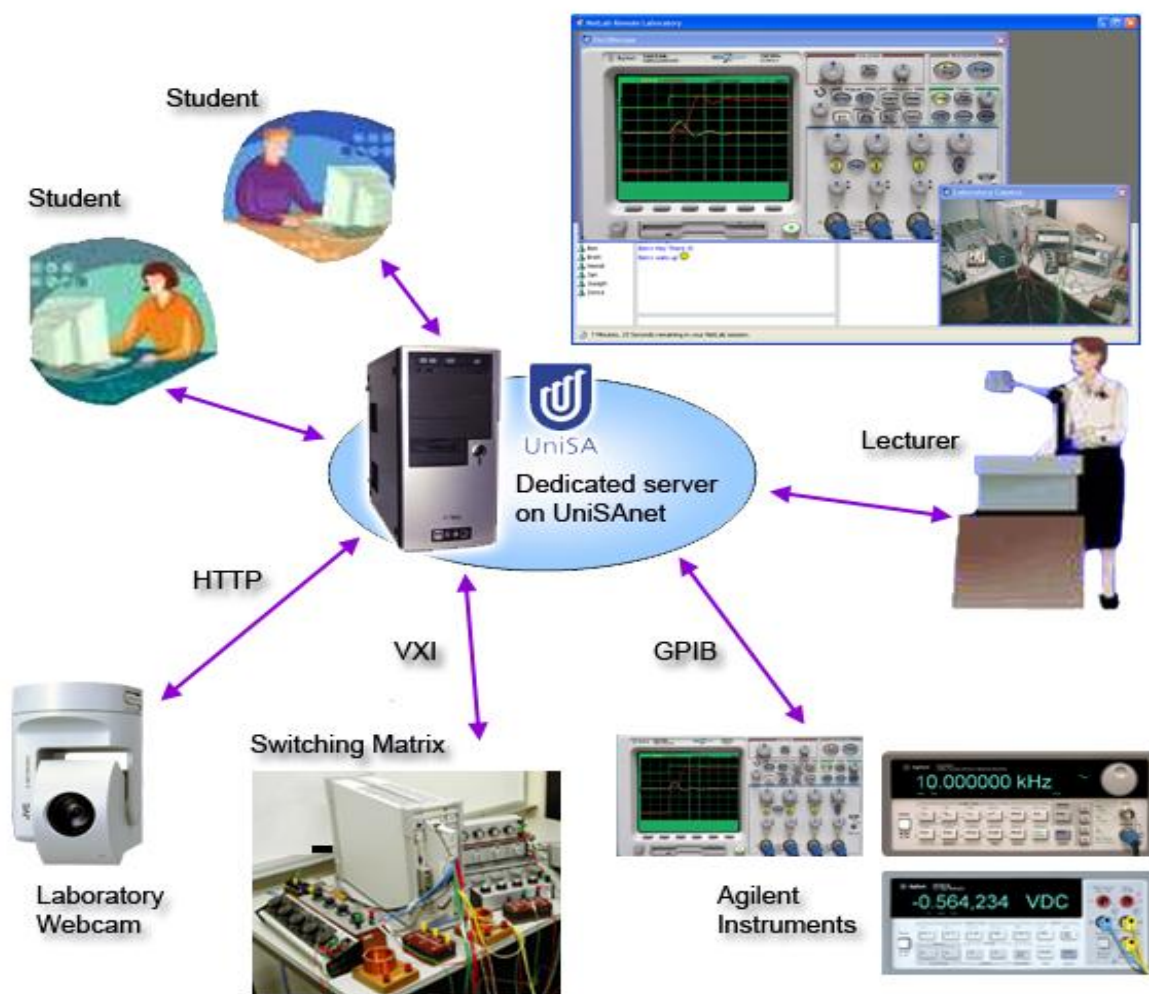


Рисунок 1.3 – Архитектура NetLab

NetLab автоматизированную предоставляет систему регистрации и авторизации пользователей, форма регистрации представлена на рисунке 1.4.

Рисунок 1.4 – Форма регистрации NetLab

Зарегистрированные пользователи должны записываться на получение доступа к лабораторным работам (рисунок 1.5)

Рисунок 1.5 – Форма записи на получения доступа NetLab

Сам интерфейс пользователя построен с использованием технологии Java WebStart, пример пользовательского интерфейса в процессе выполнения лабораторной работы представлен на рисунке 1.6

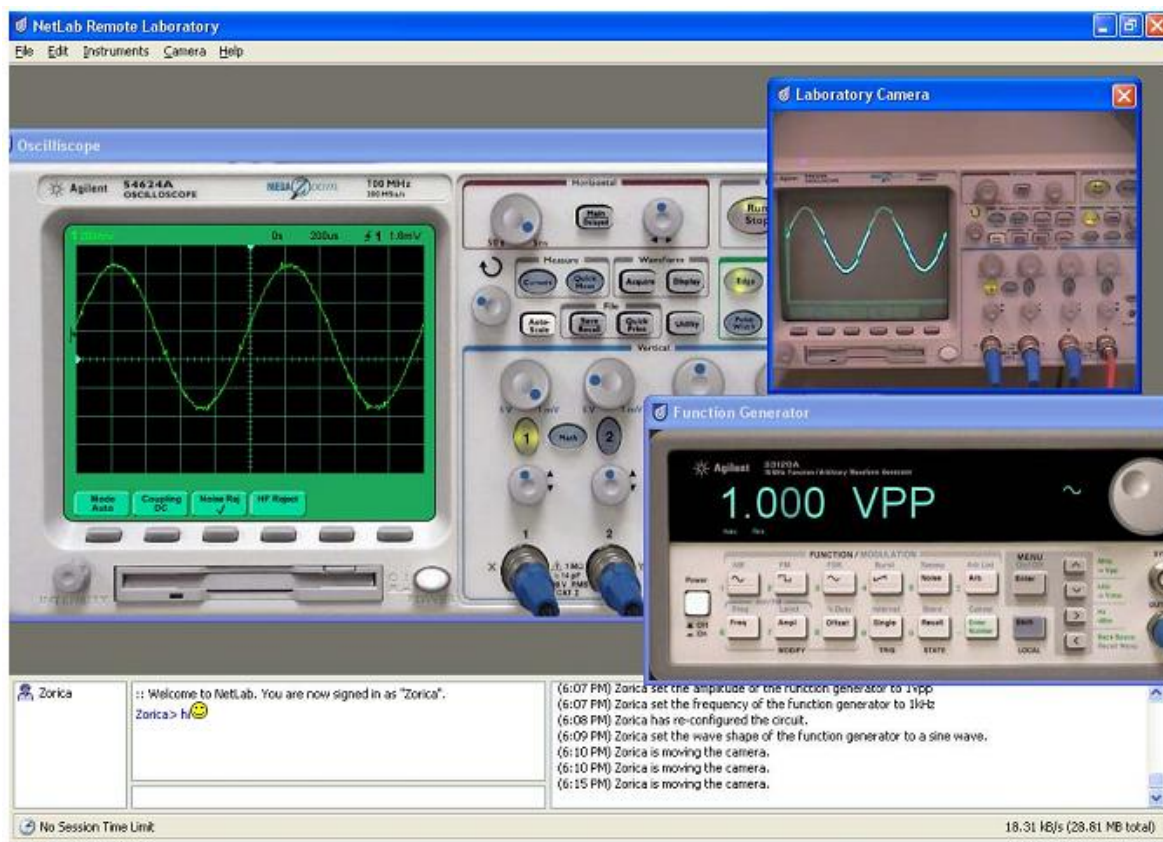


Рисунок 1.6 – Пользовательский интерфейс NetLab

1.2.2.2 Автоматизированный Лабораторный Практикум с Удаленным Доступом (АЛП УД)

Данный проект находится по адресу <http://lud.bmstu.ru> и является узко специализированной лабораторией удаленного доступа. Проект содержит четыре практикума по различным разделам курса физики (механика, электромагнетизм, квантовая физика), разработанных на кафедре "Физика" МГТУ им. Н.Э. Баумана.

В соответствии с ОСТ 9.2-98 под системой автоматизированного лабораторного практикума понимается комплекс технических, программных и методических средств, обеспечивающих автоматизированное проведение лабораторных и экспериментальных исследований непосредственно на физических объектах и (или) математических моделях. Здесь рассматриваются

только практикумы, не связанные непосредственно с математическим моделированием исследуемого объекта (их часто называют виртуальными), а выполняемые на физических стендах, где ведутся реальные эксперименты.

В настоящее время в связи с интенсивной разработкой концепции дистанционного образования большое значение приобретает автоматизированный лабораторный практикум с удаленным доступом (АЛП УД), структура которого и основные требования к его подсистемам строго регламентируются уже введенным в рамках Министерства образования РФ отраслевым стандартом ОСТ 9.2-98. Однако применение АЛП УД целесообразно не только в рамках дистанционного и открытого образования, но и при использовании традиционных очных форм проведения учебного процесса.

Задача создания и последующего коллективного использования АЛП УД с целью существенного повышения уровня практической подготовки студентов является весьма актуальной для большинства вузов Российской Федерации. Кроме того, в связи с постоянно расширяющимся использованием глобальной сети Интернет практически для любых учебных заведений открываются новые возможности доступа не только к лабораторным установкам ведущих университетов РФ, но и к уникальным стендам академических и отраслевых научных организаций.

Главная страница | АЛП УД - что это? | АЛП УД в МГТУ | Разработчики АЛП УД в МГТУ | Ссылки по АЛП УД

 МГТУ имени Н.Э. Баумана

Автоматизированный Лабораторный Практикум с Удаленным Доступом (АЛП УД)



Управление реальными экспериментальными стендами через сеть Интернет!!!

При использовании любых материалов ссылка на данный сайт обязательна

-  Интернет-лаборатория "Испытания материалов" (ИЛИМ)
-  Интернет-лаборатория "Радиотелескоп МГТУ им. Н.Э. Баумана"
-  Интернет-лаборатория "Спектрометрия плазмы и плазменные нанотехнологии"
-  Интернет - лаборатория "Робототехника"

Рисунок 1.7 – Главная страница лабораторного практикума МГТУ им Баумана

Архитектура одного из практикумов представлена на рисунке 1.8.

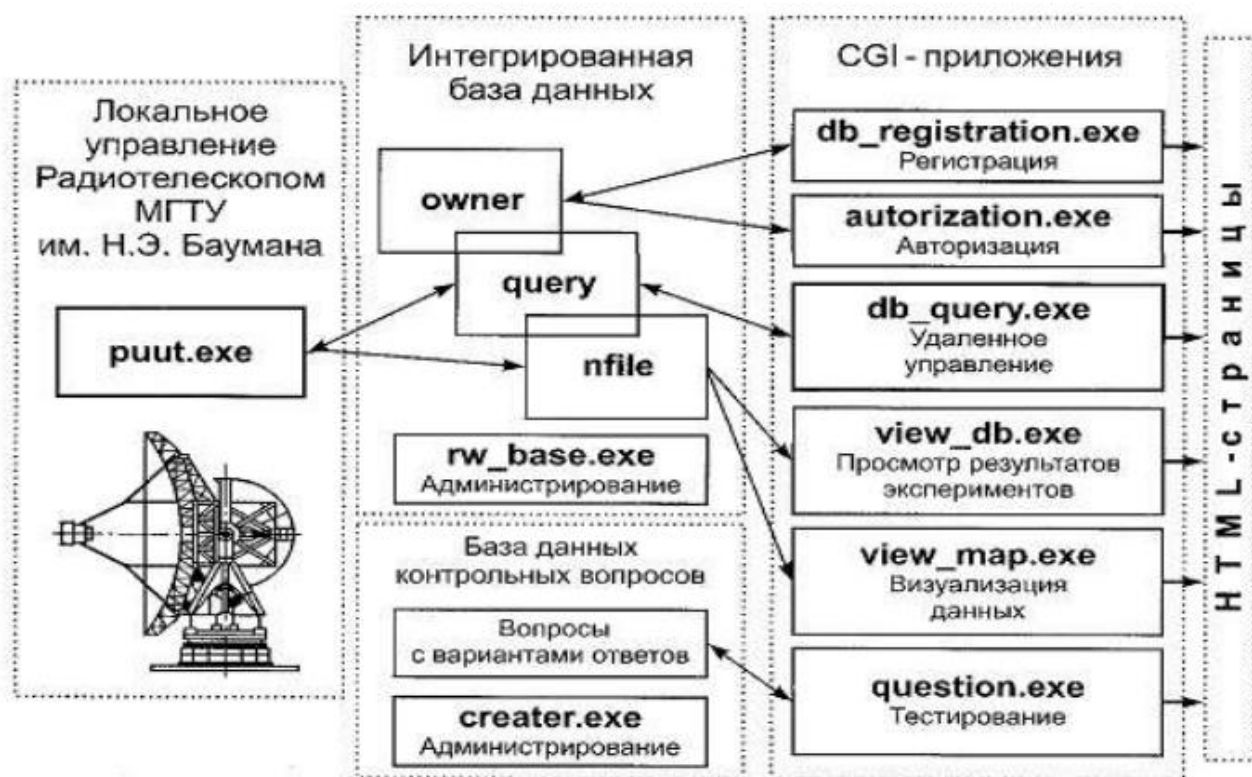



Рисунок 1.8 – Архитектура практикума

1.2.2.3 Labster

Labster (<http://www.labster.com/>) - является международно ориентированной компанией, посвященной разработке новаторских онлайн-инструментов для преподавания науки во всем мире. Основным продуктом компании является легко масштабируемая онлайн-платформа для проведения онлайн лабораторных работ


Проект содержит различные лабораторные работы, в основном виртуальные. Часть из работ являются платными и требуют покупки подписки на сервис (рисунок 1.9)

[Home](#)
[About](#)
[Work](#)
[Projects](#)
[Blog](#)
[Swimming](#)
[Programming](#)
[Articles](#)
[Shopping](#)
[Cart](#)
[Documentation](#)



Investigate the "Asian Glow" syndrome by studying the Alcohol Dehydrogenase kinetics using UV spectrophotometer.

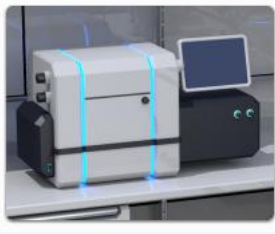
[Read More](#)
[Try it for Free!](#)
[Buy Now - 20 USD](#)



Chromatography

Separate the different compounds of drug and measure their concentrations using High Performance Liquid Chromatography (HPLC).


[Read More](#)
[Try it for Free!](#)
[Buy Now - 20 USD](#)



Next Generation Sequencing

Learn the different steps of sample preparation, sequencing, data collection and data analysis using the next generation sequencing technique.

[Read More](#)
[Try it for Free!](#)
[Buy Now - 20 USD](#)



Fermentation

Cell factory (yeast), Cell division and Bioethanol production. Fermentation technology.

[Read More](#)
[Try it for Free!](#)

Рисунок 1.9 – Примеры лабораторных работ Labster

2 Теоретическая часть

2.1 Дистанционное обучение

Дистанционное обучение это взаимодействие учителя и учащихся между собой на расстоянии, отражающее все присущие учебному процессу компоненты (цели, содержание, методы, организационные формы, средства обучения) и реализуемое специфичными средствами Интернет-технологий или другими средствами, предусматривающими интерактивность [6].

2.2 Задачи

- Разработать архитектуру системы
- Выбрать платформу для лабораторной части
- Выбрать платформу для web части
- Выбрать способ взаимодействия между web и лабораторными частями
- Подобрать виртуальные приборы для тестирования

2.3 Разработка архитектуры системы

Учитывая что необходимые компоненты лаборатории удаленного доступа в данном случае это: Программное обеспечение для создания виртуальных приборов (либо реальные приборы и программные интерфейсы к ним) и ПО для обеспечения удаленного доступа, архитектура виртуальной лаборатории представлена в простейшем виде на рисунке 2.1

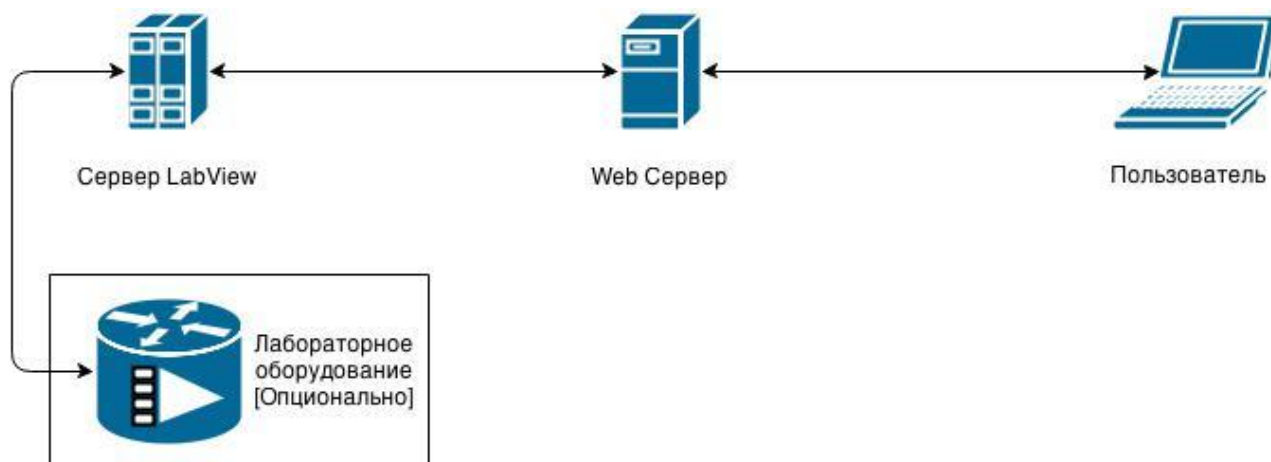


Рисунок 2.1 – Архитектура виртуальной лаборатории

Соединение между сервером LabVIEW и Web сервером в данном случае может быть реализовано несколькими способами, сравнительная характеристика которых представлена в таблице 2.1.

Прибор как Web-сервис	VI публикуется на встроенном Web Application сервере LabVIEW	+: быстроедействие, может использоваться любым ПО -: вывод только текстовой информации, обновление вывода только по HTTP запросу
LabVIEW remote panels	VI публикуется на встроенном Remote Panel сервере LabVIEW, после чего может быть встроен в любой HTML документ и открыт браузером при наличии LabView RuntimeEnvironment	+: Отображение полнофункциональной панели прибора в браузере -: В случае одновременного использования прибора контролировать прибор может только 1 из пользователей
Реализовать обработку соединений в приборе	LabVIEW представляет полный набор компонент для построения своего серверного приложения внутри VI	+: Гибкость -: Трудоемкость, усложняет разработку самих VI

Таблица 2.1 – Сравнительная характеристика способов соединения

2.4 Разработка виртуальной лаборатории

2.4.1 Выбор платформы для виртуальной лаборатории

Доступные программные платформы позволяющие создавать виртуальные приборы или интерфейсы к реальным приборам перечисленно в таблице 2.2

Comedi	OpenSource коллекция драйверов для популярных приборов под операционные системы семейства Linux
MyOpenLab	OpenSource ПО позволяющее создавать различные программные блоки и соединять их друг с другом, потенциально подходит для создания виртуальных приборов. Доступно только на немецком языке
LabVIEW	Среда разработки и платформа для выполнения программ, созданных на графическом языке программирования «G». Подходит для создания виртуальных приборов и взаимодействия с реальными приборами.

Таблица 2.2 – Программные платформы для лабораторной части проекта

Как видно из таблицы наиболее полный комплект инструментов предоставляет платформа LabVIEW, которая рассмотрена подробнее далее.

2.4.2 Платформа LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) представляет собой платформу и среду разработки для визуального языка программирования от National Instruments. Графический язык назван «G» и первоначально был выпущен для Apple Macintosh в 1986 году.

LabVIEW обычно используется для сбора данных, управления приборами и промышленной автоматизации на различных платформах, включая Microsoft Windows, различных версий UNIX, Linux и Mac OS X.

Язык программирования, используемый в LabVIEW является языком поточного программирования данных. Исполнение определяется структурой графической блок-схемы, на которой программист соединяет различные функции-узлы, рисуя провода. Эти провода могут передавать значения переменных на любой узел, который может выполнять различные операции, как только все его входные данные становятся доступными. G по своей природе пригоден для параллельного программирования, так как данные могут одновременно быть доступны нескольким узлам. Multi-обработка и аппаратная многопоточность используется автоматически с помощью встроенного планировщика, который мультиплексирует несколько потоков ОС по узлам готовых к исполнению.

LabVIEW связывает создание пользовательских интерфейсов (названных передними панелями) в цикл развития. Программы/подпрограммы LabVIEW называют виртуальными инструментами (ВИ). У каждого ВИ есть три компонента: блок-схема, передняя панель и панель соединений. Последняя используется, чтобы представлять ВИ в блок-схемах других ВИ. Передняя панель строится, используя средства управления и индикаторы. Средства управления - входы – позволяют пользователю предоставлять информацию к ВИ. Индикаторы - выходы – они отображают, результаты, основанные на входных данных ВИ. Задняя панель, которая является блок-схемой, содержит графический исходный код. Все объекты, помещенные в переднюю панель, появятся на задней панели как терминалы. Задняя панель также содержит структуры и функции, которые выполняют операции на средствах управления и снабжают данными к индикаторам. Структуры и функции найдены на палитре функций и могут быть помещены на заднюю панель. Коллективно средства управления, индикаторы, структуры и функции называются узлами (nodes). Узлы связаны с друг другом используя провода. Таким образом ВИ можно или управлять как программой, с передней панелью, служащей пользовательским интерфейсом, или программно, когда помещена как узел на блок-схему. Это подразумевает, что каждый ВИ может быть легко проверен прежде чем быть включенным как подпрограмма в большую программу.

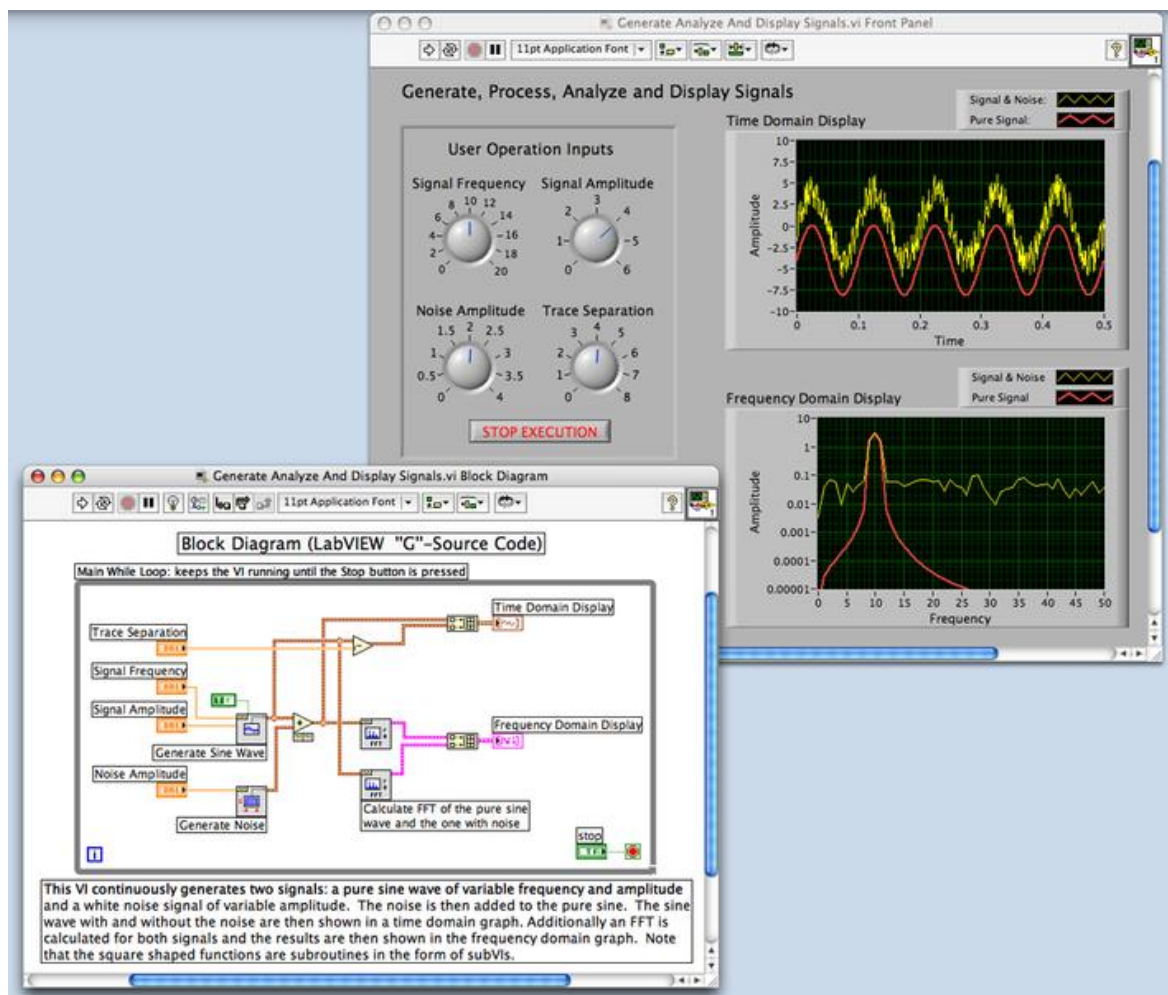


Рисунок 2.2 – Виртуальный инструмент LabVIEW

Графический подход также позволяет "непрограммистам" создавать программы, визуально.

Также LabVIEW поддерживает огромный спектр оборудования различных производителей и имеет в своём составе (либо позволяет добавлять к базовому пакету) многочисленные библиотеки компонентов:

- для подключения внешнего оборудования по наиболее распространённым интерфейсам и протоколам (RS-232, GPIB-488, TCP/IP и пр.);
- для удалённого управления ходом эксперимента;
- для управления роботами и системами машинного зрения;
- для генерации и цифровой обработки сигналов;
- для применения разнообразных математических методов обработки данных;
- для визуализации данных и результатов их обработки (включая 3D-модели);
- для моделирования сложных систем;

- для хранения информации в базах данных и генерации отчетов;
- для взаимодействия с другими приложениями в рамках концепции COM/DCOM/OLE.

Специальный компонент LabVIEW Application Builder позволяет создавать LabVIEW-программы, пригодные для выполнения на тех компьютерах, на которых не установлена полная среда разработки. Для работы таких программ требуется бесплатно распространяемый компонент «LabVIEW Runtime Engine» и, при необходимости, драйверы используемых внешних устройств.

2.4.3 Технология Remote Panel

Несколько версий LabVIEW содействовали способности разрабатывать распределенные приложения: TCP/IP, интернет-инструментов, VI Серверов, Веб-публикаций Передней панели, Remote Data Acquisition (RDA), DataSocket, и так далее. Кроме того, несколько сторонних наборов инструментов позволили основанный на контроле приборов через интернет: LabVNC и AppletVIEW. Также конечно, всегда был и решения для ПК которые обеспечивают общее дистанционное управление.

Вы можете создать распределенные приложения, используя эти инструменты. Однако каждый представляет собой уникальные проблемы, часто требуя передовых программных методов и разработки таможенных механизмов обработки данных.

Поэтому был разработан LabVIEW remote panels. Одна из первых вещей, которые Вы изучаете, когда Вы начинаете программировать в LabVIEW, - то, что ВИ состоят из передних панелей, диаграмм, панели подключений.

С LabVIEW Вы можете использовать свою переднюю панель на машине, которая является отдельной от того, где эти ВИ проживают и выполняются. Кроме того, Вы можете включить переднюю панель в веб-страницу и управлять ею в пределах той страницы. Все, что требуется на машине клиента для выполнения в веб-странице, является браузером и двигателем времени выполнения LabVIEW и программным расширением браузера.

Вы будете поражены тем, как легко формировать и использовать такие приборы с использованием Remote Panels. Необходимо всего два шага:

- Разрешить Web-сервер LabVIEW на машине сервера.
- Соединиться и запустить remote panels по машине клиента.

Схема работы Remote Panels представлена на рисунке 2.3.

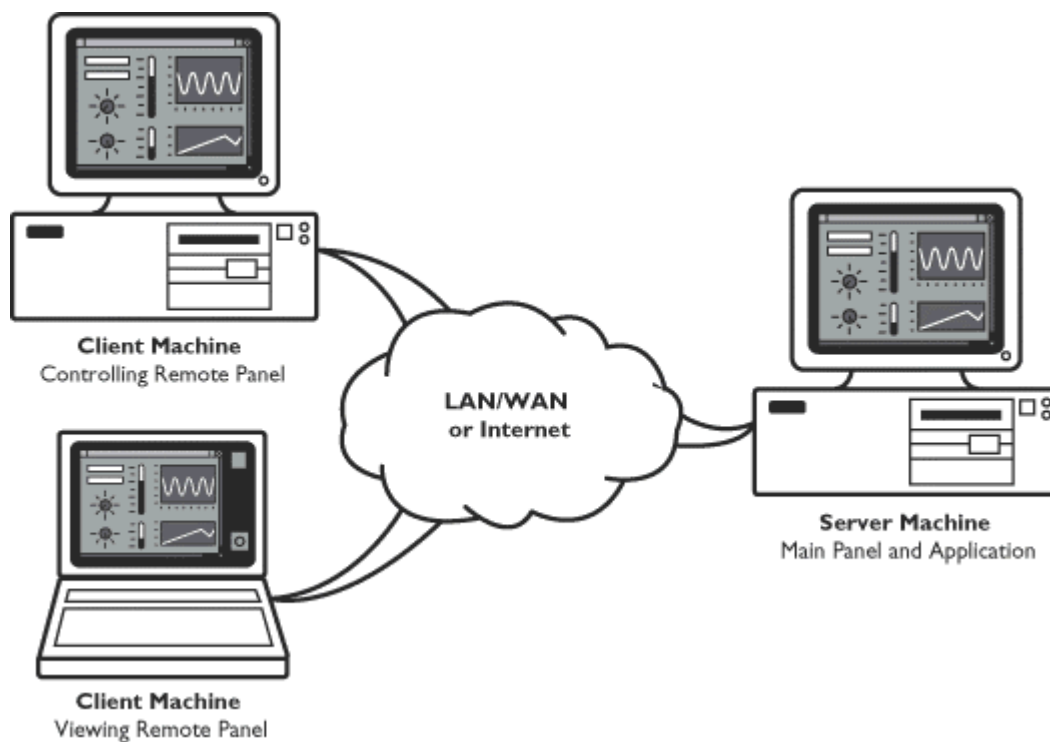


Рисунок 2.3 – Схема работы LabVIEW Remote Panels

2.4.4 LabVIEW Runtime Engine

LabVIEW Runtime Engine представляет из себя набор библиотек позволяющих выполнять приложения LabVIEW на компьютерах без установленного LabVIEW.

2.4.5 Обзор виртуальных приборов

2.4.5.1 Heat equation for a metal plate

Данный виртуальный прибор LabVIEW моделирует процесс нагревания металлической пластинки при известных начальных температурах на её границах. Передняя панель прибора представлена на рисунке 2.4.

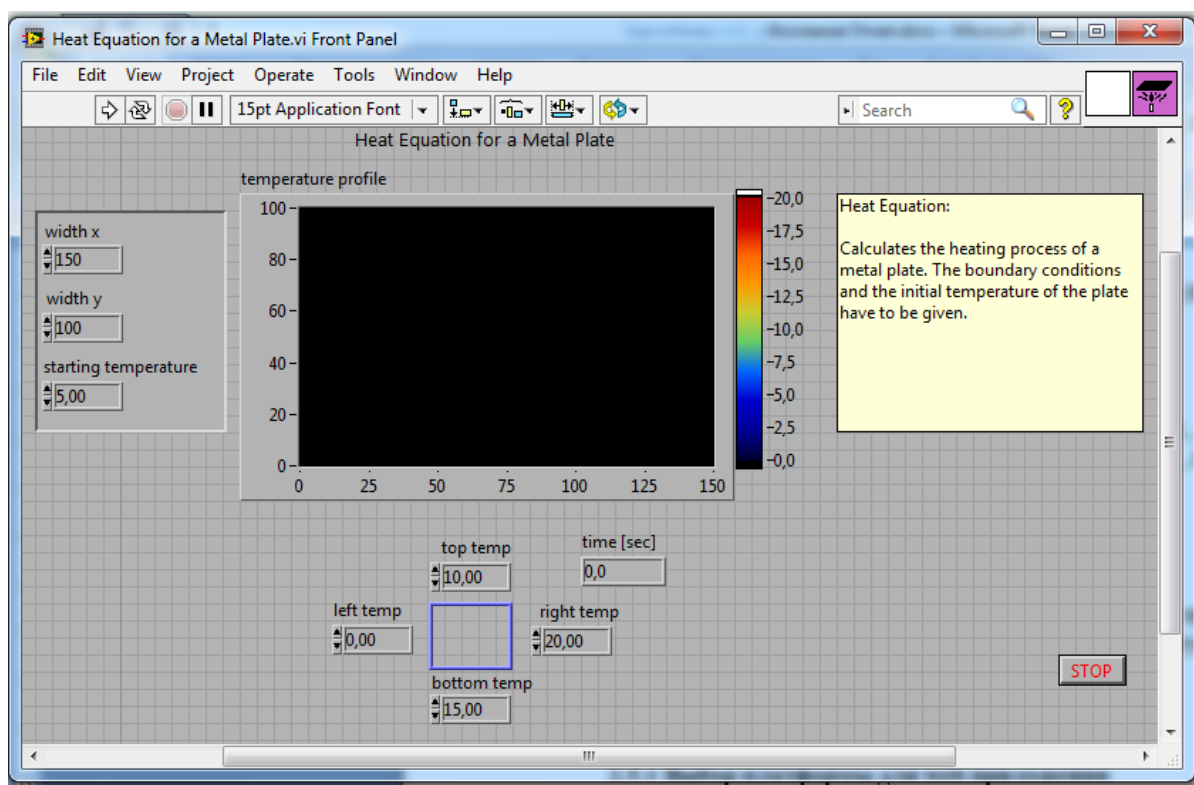


Рисунок 2.4 – Передняя панель прибора

Блок схема прибора представлена на рисунке 2.5.

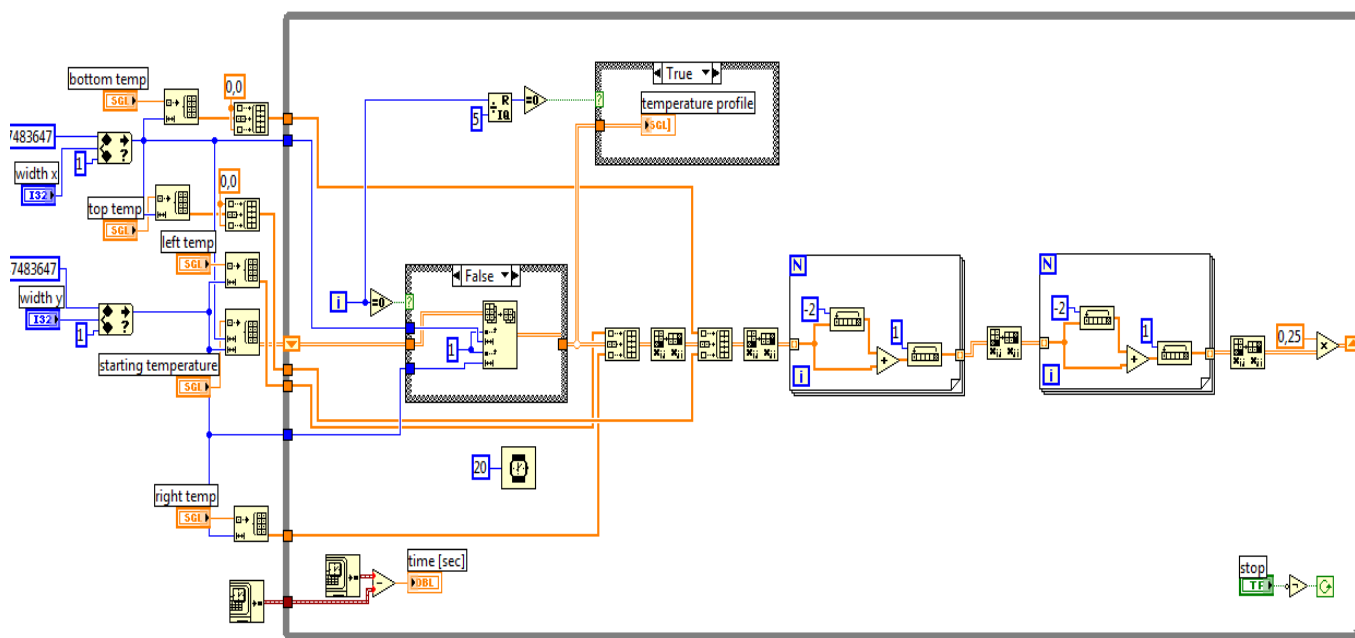


Рисунок 2.5 – Блок схема прибора

2.4.5.2 Исследование 1-D функции

Данный прибор позволяет строить и изучать графики заданных одномерных функций, включая графики их дифференцирования и интегрирования, изучать экстремумы функций, находить корни.

Приборная панель прибора представлена на рисунке 2.6

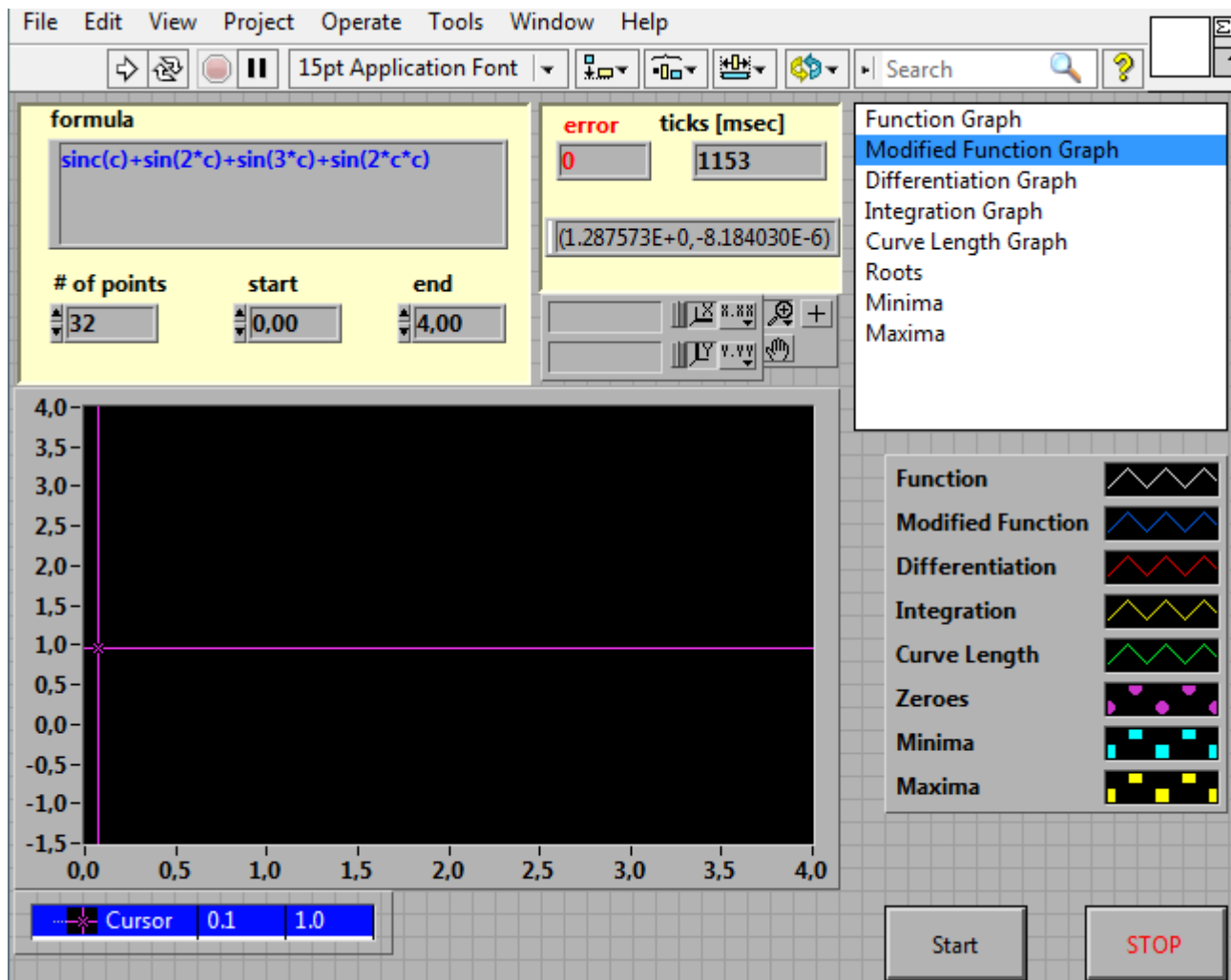


Рисунок 2.6 – Передняя панель прибора

Блок схема прибора представлена на рисунке 2.7

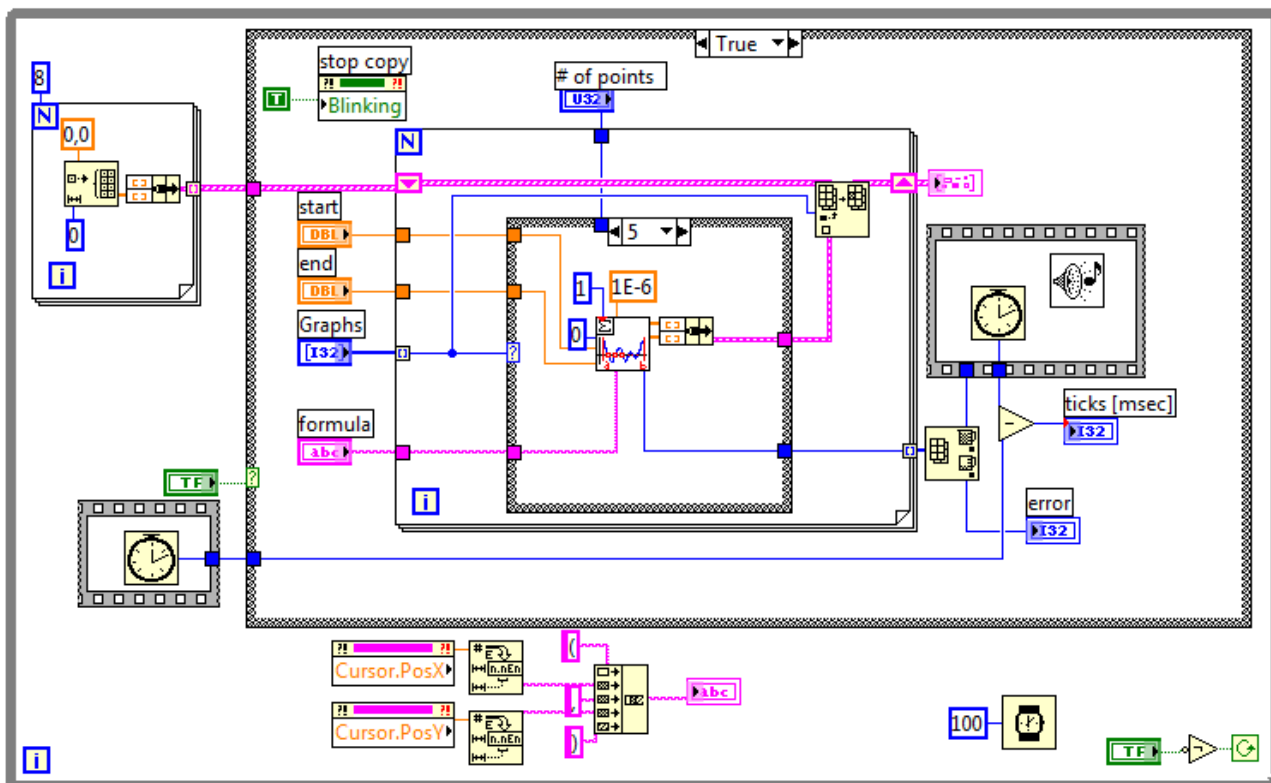


Рисунок 2.7 – Блок схема прибора

2.5 Разработка web приложения

2.5.1 Выбор платформы для web приложения

На сегодняшний день существует довольно много платформ и фреймворков позволяющих разработать web приложение – php, java, python, ruby on rails и так далее. Все они пригодны для создания web проектов любой сложности.

Для создания данного проекта лаборатории удаленного доступа было решено использовать язык программирования Ruby и фреймворк Rails, ввиду того что эта комбинация сейчас набирает популярность на рынке web программирования, хорошо документирована, регулярно обновляется разработчиками и является бесплатной.

2.5.2 Язык Ruby

Ruby динамический, рефлексивный, интерпретируемый высокоуровневый язык программирования для быстрого и удобного объектно-ориентированного программирования. Язык обладает независимой от операционной системы реализацией многопоточности, строгой динамической типизацией, сборщиком мусора и многими другими возможностями. Ruby близок

по особенностям синтаксиса к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и Клу.

Кроссплатформенная реализация интерпретатора языка является полностью свободной.

Язык следует принципу «наименьшей неожиданности»: программа должна вести себя так, как ожидает программист. Однако в контексте Ruby это означает наименьшее удивление не при знакомстве с языком, а при его основательном изучении. Сам Мацумото утверждает, что целью разработки была минимизация неожиданностей при программировании для него, но после распространения языка он с удивлением узнал, что мышление программистов похоже, и для многих из них принцип «наименьшей неожиданности» совпал с его принципом.

Ruby также унаследовал идеологию языка программирования Perl в части предоставления программисту возможностей достижения одного и того же результата несколькими различными способами. Люди различны, и им для свободы необходима возможность выбирать. «Я предпочитаю обеспечить много путей, если это возможно, но поощрять или вести пользователей, чтобы выбрать лучший путь, если это возможно».

Одной из основных целей разработки было освобождение программистов от рутинной работы, которую вычислитель может выполнять быстрее и качественнее. Особое внимание, в частности, уделялось будничным рутинным занятиям (обработка текстов, администрирование), и для них язык настроен особенно хорошо.

В противовес машинно-ориентированным языкам, работающим быстрее, целью этой разработки был язык, наиболее близкий к человеку. Любая работа с компьютером выполняется людьми и для людей, и необходимо заботиться в первую очередь о затрачиваемых усилиях людей. Язык позволяет максимально быстро и просто для человека выполнить задачу, хотя, возможно, это и потребует дополнительного времени работы компьютера.

Принципы программирования и устройства языка иногда выделяются в термин «Путь Ruby» (англ. Ruby Way). Хэл Фултон выделяет такие принципы, как «просто, но не слишком просто», «принцип наименьшего удивления», вторичность скорости работы программы, динамичность, простые строгие правила, выполнение которых не доходит до педантизма, потребность создавать полезные и красивые программы как причина программирования. В целом они не имеют точной формулировки и иногда этот термин используется для критики.

Возможности языка:

- Имеет лаконичный и простой синтаксис, частично разработанный под влиянием Ада, Eiffel и Python.
- Позволяет обрабатывать исключения в стиле Java и Python.

- Позволяет переопределять операторы, которые на самом деле являются методами.
- Полностью объектно-ориентированный язык программирования. Все данные в Ruby являются объектами в понимании Smalltalk. Единственное исключение — управляющие конструкции, которые в Ruby в отличие от Smalltalk не являются объектами. Например, число «1» — это экземпляр класса Fixnum. Также поддерживается добавление методов в класс и даже в конкретный экземпляр во время выполнения программы.
- Не поддерживает множественное наследование, но вместо него может использоваться концепция «примесей», основанная в данном языке на механизме модулей.
- Содержит автоматический сборщик мусора. Он работает для всех объектов Ruby, в том числе для внешних библиотек.
- Создавать расширения для Ruby на Си очень просто частично из-за сборщика мусора, частично из-за несложного и удобного API.
- Поддерживает замыкания с полной привязкой к переменным.
- Поддерживает блоки кода (код заключается в { ... } или do ... end). Блоки могут использоваться в методах или преобразовываться в замыкания.
- Целые переменные в Ruby автоматически конвертируются между типами Fixnum (32-разрядные) и Bignum (больше 32 разрядов) в зависимости от их значения, что позволяет производить целочисленные математические расчёты со сколь угодно большой точностью.
- Не требует предварительного объявления переменных, но для интерпретатора желательно, чтобы переменным присваивалось пустое значение nil (тогда интерпретатор знает, что идентификатор обозначает переменную, а не имя метода). Язык использует простые соглашения для обозначения области видимости. Пример: просто var — локальная переменная, @var — переменная экземпляра (член или поле объекта класса), @@var — переменная класса, \$var — глобальная переменная.
- В Ruby непосредственно в языке реализованы многие шаблоны проектирования, так, например, «одиночка» (singleton) может быть (хотя и не обязан) реализован добавлением необходимых методов к одному конкретному объекту (см. ниже).
- Может динамически загружать расширения, если это позволяет операционная система.
- Имеет независимую от ОС поддержку невытесняющей многопоточности.
- Перенесён на множество платформ. Он разрабатывался на Linux, но работает на многих версиях Unix, DOS, Microsoft Windows (в частности, Win32), Mac OS, BeOS, OS/2 и т. д.

2.5.3 Платформа Rails

Ruby on Rails это фреймворк, написанный на языке программирования Ruby. Ruby on Rails предоставляет архитектурный образец Model-View-Controller (модель-представление-контроллер) для веб-приложений, а также обеспечивает их интеграцию с веб-сервером и сервером базы данных.

Ruby on Rails является открытым программным обеспечением и распространяется под лицензией MIT.

Ruby on Rails является общедоступной структурой веб-приложения, которая бежит через язык программирования Ruby. Это - структура полного стека: это позволяет создавать страницы и заявления, которые собирают информацию от веб-сервера, говорят или подвергают сомнению базу данных и отдают шаблоны из коробки. В результате Rails показывают систему направления, которая независима от веб-сервера.

Ruby on Rails подчеркивает использование известных паттернов программирования, и принципы, такие как active record pattern, convention over configuration (CoC), don't repeat yourself (DRY), and model-view-controller (MVC).

Основными компонентами приложений Ruby on Rails являются модель (англ. model), представление (англ. view) и контроллер (англ. controller). Ruby on Rails использует REST-стиль построения веб-приложений.

Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (таких как каталог продуктов или список заказов). Объекты модели могут осуществлять загрузку и сохранение данных в реляционной базе данных, а также реализуют бизнес-логику.

Для хранения объектов модели в реляционной СУБД по умолчанию в Rails 3 использована библиотека ActiveRecord. Конкурирующий аналог — DataMapper. Существуют плагины для работы с нереляционными базами данных, например Mongoid для работы с MongoDB.

Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на манипуляцию данными в контроллер (как правило, представление не изменяет непосредственно модель).

В Ruby on Rails представление описывается при помощи шаблонов ERB. Они представляют собой файлы HTML с дополнительными включениями фрагментов кода Ruby (Embedded Ruby или ERb). Вывод, сгенерированный встроенным кодом Ruby, включается в текст шаблона, после чего получившаяся страница HTML возвращается пользователю. Кроме ERB возможно использовать ещё около 20 шаблонизаторов, в том числе Haml.

Контроллер в Rails — это набор логики, запускаемой после получения HTTP-запроса сервером. Контроллер отвечает за вызов методов модели и запускает формирование представления.

Соответствие интернет-адреса с действием контроллера (маршрут) задается в файле config/routes.rb.

Контроллером в Ruby on Rails является класс, наследованный от ActionController::Base. Открытые методы контроллера являются так называемыми действиями (actions). Action часто соответствует отдельному представлению. Например, по запросу пользователя admin/list будет вызван метод list класса AdminController и затем использовано представление list.html.erb. Модель MVC представлена на рисунке 2.2.



Рисунок 2.2 – Модель MVC

Предпочтительным методом интеграции с веб-серверами является проксирование — использование веб-сервера в качестве прокси-сервера перед сервером приложения. Особняком стоят модули Phusion Passenger для интеграции с серверами Apache и nginx.

Ruby on Rails использует интерфейс RACK, что позволяет использовать менее распространённые механизмы (FCGI, CGI, SCGI). Ruby on Rails может работать с Apache, Lighttpd или любым другим веб-сервером, поддерживающим FastCGI. Для разработки и отладки часто используется веб-сервер WEBrick, встроенный в Ruby, или Mongrel. В качестве сервера базы данных

поддерживаются MySQL, Firebird, PostgreSQL, DB2, Oracle и Microsoft SQL Server. Также поддерживается встраиваемая база данных SQLite.

Для Windows существует дистрибутив Instant Rails с настроенной и готовой к работе сразу после установки рабочей средой для разработки Rails-приложений, которая включает в себя сервер Apache и СУБД MySQL. Для платформ Windows, Linux, Mac OS X имеется комплексный установщик BitNami RubyStack[5], включающий в себя все необходимое для разработки в среде Rails, включая Ruby, RubyGems, Ruby on Rails, MySQL, Apache, Mongrel и Subversion.

Помимо этого сайты BitNami.org и JumpBox.com бесплатно предлагают образы VMware с готовой Linux-средой для развертывания RoR-приложений. Эти образы можно подключить к своему серверу виртуальных машин или развернуть в предлагаемой облачной среде.

Для разработки AJAX-приложений в RoR по умолчанию используется javascript-фреймворк jQuery, однако вместо него можно использовать и другие библиотеки. В ранних версиях Ruby on Rails (до 3.1), js-фреймворком по умолчанию был Prototype.

2.5.4 Система управления контентом Comfortable Mexican Sofa

ComfortableMexicanSofa - мощная CMS для Rails 4

CMS (Content management system) - информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым). Основные функции CMS:

- Предоставление инструментов для создания содержимого, организация совместной работы над содержимым,
- Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.,
- Публикация содержимого,
- Представление информации в виде, удобном для навигации, поиска.

В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии, номера телефонов, научные данные и так далее. Такая система часто используется для хранения, управления, пересмотра и публикации документации. Контроль версий является одним из основных её преимуществ, когда содержимое изменяется группой лиц.

Большинство современных CMS имеют модульную архитектуру, что позволяет администратору, самому выбирать и настраивать те компоненты, которые ему необходимы. Типичные модули: динамическое меню, блог, новости, опросы, поиск по сайту, статистика посещений, гостевая книга и т. д.

Сайты организованные посредством системы управления контентом основаны на следующих технологиях: веб-сервер, хранилище данных (зачастую СУБД, например такие как MySQL или PostgreSQL, однако существуют и noSQL CMS), веб-приложение для обеспечения работы самой системы, визуальный (WYSIWYG) редактор страниц, файловый менеджер с веб-интерфейсом для управления файлами сайта, система управления правами пользователей и редакторов сайта.

Существуют разнообразные системы управления сайтом, среди которых встречаются платные и бесплатные, построенные по разным технологиям. Каждый сайт имеет панель управления, которая является только частью всей программы, достаточной для управления сайтом.

Наиболее распространены следующие технологические платформы используемые в качестве основы веб-приложения, реализующего работу CMS: PHP, Perl, .NET.

Существует термин контент-менеджер, обозначающий род профессиональной деятельности — редактор сайта или сотрудника, работающего с CMS.

Большая часть современных систем управления содержимым реализуется в виде визуального (WYSIWYG) редактора — программы, которая создаёт HTML-код из специальной упрощённой разметки, позволяющей пользователю проще форматировать текст.

Особенности ComfortableMexicanSofa:

- Простая интеграция с Rails 4 приложения
- Позволяет создавать свое приложение в Rails, не в CMS
- Мощный механизм templating'a страниц
- Много сайтов управляются единственной установкой
- Многоязычная Поддержка (i18n) (cs, da, de, en, es, fr, ja, nl, ru, pt-BR, sv, zh-CN)
- Приспособления для начального заполнения содержания
- История ревизий
- Настраиваемый admin интерфейс

2.5.5 Система авторизации Devise

Devise - гибкое решение для идентификации для Rails, основанное на Warden. Это:

- Базируется на Rack;
- Полное решение MVC, основанное на движке Rails;
- Позволяет одновременно регистрировать и авторизовать несколько моделей.

- Основано на понятии модульности: используйте, в чем вы действительно нуждаетесь.

Состоит из 10 модулей:

- Database Authenticatable: шифрует и хранит пароль в базе данных, чтобы проверить подлинность пользователя, регистрируясь. Идентификация может быть сделана оба посредством POST запросов или Базовой аутентификации HTTP.

- Omniauthable: добавляет Омниот (<https://github.com/intridea/omniauth>) поддержку.

- Confirmable: посылает электронные письма с инструкциями по подтверждению и проверяет, подтвержден ли счет уже во время, регистрируются.

- Recoverable: изменяет пользовательский пароль и отправляет указания сброса.

- Registerable: пользователи подписания ручек посредством процесса регистрации, также позволяя им отредактировать и разрушить их счет.

- Rememberable: управляет производством и прояснением символа для запоминания пользователя от спасенного печенья.

- Trackable: следы подписываются в количестве, метках времени и IP-адресе.

- Timeoutable: истекает сессии, у которых нет деятельности в установленный период времени.

- Validatable: обеспечивает проверки электронной почты и пароля. Это дополнительное и может быть настроено, таким образом, Вы в состоянии определить свои собственные проверки.

- Lockable: блокирует аккаунт после того, как конкретное количество неудавшихся попыток было совершено. Может открыть по электронной почте или после указанного периода времени.

2.5.6 ZURB Foundation framework

ZURB Foundation представляет собой бесплатный набор инструментов для создания веб-сайтов и веб-приложений. Он содержит HTML и CSS на основе шаблонов дизайна для типографии, форм, кнопок, навигации и других элементов интерфейса, а также дополнительных расширений JavaScript.

Он используется в Dictionary.com, веб-приложениях ZURB, и т.д.

Foundation возник как проект ZURB разработки кльбой делает интерфейсный код быстрее и лучше. В октябре 2011 года ZURB выпущен Foundation 2.0 в качестве открытым исходным кодом под лицензией MIT. В июне 2012 ZURB выпустила крупное обновление, Фонд 3.0. В феврале 2013 года ZURB выпустила еще один крупное обновление, Фонд 4.0. В ноябре 2013 ZURB выпустила еще один крупное обновление, Фонд 5.0.

Foundation имеет относительно неполную поддержку HTML 5 и CSS 3, но она совместима со всеми основными браузерами. Базовая информация о совместимости сайтов или приложений доступен для всех устройств и браузеров. Существует понятие частичной совместимости, что делает основная информация о сайте доступны для всех устройств и браузеров. Например, свойства, введенные в CSS3 для закругленными углами, градиенты и тени используются Foundation, несмотря на отсутствие поддержки со стороны старших веб-браузеров. Они расширяют функциональные возможности инструментария, но не являются обязательными для его использования.

Начиная с версии 2.0 она также поддерживает реагировать конструкцию. Это означает, что графический дизайн веб-страниц регулирует динамически, с учетом характеристик используемого устройства (ПК, планшеты, мобильный телефон). Кроме того, начиная с 4.0 оно приняло мобильного первая подход, проектирования и разработки для мобильных устройств сначала, и повышения веб-страниц и приложений для больших экранов.

Foundation является открытым исходным кодом и предоставляется в GitHub. Разработчикам предлагается участвовать в проекте и сделать свой вклад в платформу.

Foundation является модульной и состоит в основном из серии Sass стилей, которые реализуют различные компоненты инструментария. Компонент стилей могут быть включены с помощью Sass или настроив начальную загрузку Foundation. Разработчики могут адаптировать сам файл Foundation, выбора компонентов, которые они хотят использовать в своем проекте.

Корректировки возможны в ограниченном объеме с помощью центральных стилей конфигурации. Более глубокие изменения возможны, изменив переменные Sass.

Использование Sass языке стилей позволяет использовать переменных, функций и операторов, вложенных селекторов, а также так называемые Mixins.

Начиная с версии 3.0, конфигурация Foundationa также есть специальная опция "Настроить" в документации. Более того, разработчики используют в форме выбрать желаемые компоненты и настроить, при необходимости, значения различных вариантов с их потребностями. Впоследствии генерируемый пакет уже включает в себя предварительно построенный CSS стилей.

Foundation поставляется со стандартной сеткой, шириной 940 пикселей. Инструментарий полностью реагировать на основе использования различных разрешений и типов устройств: мобильных телефонов, портретной и альбомной ориентации, планшеты и ПК с низким и высоким разрешением (широкоэкранный). Это автоматически настраивает ширину столбцов.

Foundation предоставляет набор стилей, которые обеспечивают основные определения стиля для всех ключевых компонентов HTML. Они обеспечивают

браузер и общесистемного форму, современный внешний вид для форматирования текста, таблиц и элементов формы.

В дополнение к регулярным элементам HTML, Foundation содержит другие обычно используемые элементы интерфейса. К ним относятся кнопки с расширенными функциями (например, группировка кнопок или кнопок с возможностью выпадающего, делают и навигационные списки, горизонтальные и вертикальные вкладки, навигации, цепочку переходов, разбиения на страницы и т.д.), этикетки, передовые типографские возможности и форматирование для

Компоненты JavaScript из Foundationa основаны на Zepto.js, более легкой, но API-совместимый альтернативы рамках JQuery JavaScript. Они обеспечивают дополнительные элементы пользовательского интерфейса, такие как диалоговые, всплывающих подсказок и каруселями. Они также расширить функциональность некоторых существующих элементов интерфейса, в том числе, например, сплит кнопку раскрывающегося списка. В версии 4.0 поддерживаются следующие плагины JavaScript: оповещения, очистка, куки, выпадающие, формы, roll-over, Magellan, orbit, filler, tips, и topbar.

2.5.7 СУБД PostgreSQL

PostgreSQL является мощной объектно-реляционной системой управления базами данных с открытым исходным кодом. Он имеет более чем 15-летний стаж активного развития и проверенной архитектуры, что снискало ему репутацию надежности, целостности данных и корректности. PostgreSQL работает на всех основных операционных системах, в том числе Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), и Windows. Он полностью ACID совместимый, имеет полную поддержку внешних ключей, объединений, представлений, триггеров и хранимых процедур (на нескольких языках). Он включает в себя большую часть SQL: 2008 типы данных, в том числе INTEGER, числовые, логические, CHAR, VARCHAR, DATE, интервал, а TIMESTAMP. Он также поддерживает хранение больших двоичных объектов, в том числе фотографии, звуки или видео. Он имеет собственные программные интерфейсы для C / C ++, Java, . Net, Perl, Python, Ruby, Tcl, ODBC, среди прочего, и исключительной документации.

База данных корпоративного класса PostgreSQL может похвастаться сложным функционалом, таким как Multi-Version управление параллелизмом (MVCC), восстановление по точкам отката, табличная асинхронная репликация, вложенные транзакции (точки сохранения), горячие резервные копии, сложный планировщик запросов и оптимизатор, журналирование на случай поломки. Он поддерживает международные наборы символов и многобайтовые кодировки Unicode, их сортировки, чувствительность к регистру, и форматирование. Он масштабируемо как в смысле количества данных, так и по количеству

одновременно работающих пользователей. Существуют активные системы PostgreSQL в производственной среде, которые управляют более 4 терабайтами данных.

PostgreSQL гордится тем, что соответствует стандартам. Его реализация SQL сильно соответствует ANSI-SQL: 2008. Он имеет полную поддержку вложенных запросов (включая выбор ECOM) и уровни изоляции сериализуемых транзакций. Также PostgreSQL имеет полностью реляционный системный каталог, поддерживающий множество схем баз данных, его каталог также доступен посредством информационной схемы, как это определено в стандарте SQL.

Обеспечения целостности данных включают (Compound) первичные ключи, внешние ключи с поддержкой запрета и каскадирования изменений / удалений, ограничения проверки, ограничения уникальности и не нулевые ограничения.

Он также имеет множество расширений и улучшений. Среди удобств являются автоинкрементные последовательности, LIMIT / OFFSET, позволяющие возвращать результирующий набор только частично. PostgreSQL поддерживает составные, уникальные, частичные и функциональные индексы, которые могут использовать любой из видов деревьев: B-дерева, R-дерева, хэш, или методов хранения GiST.

GiST (Обобщенная дерево поиска) индексации представляет собой развитую систему, которая объединяет широкий набор различных алгоритмов сортировки и поиска, включая B-дерева, B +-дерева, R-дерева, деревья частичных сумм, упорядоченных B +-деревьев и многие другие. Он также предоставляет интерфейс, обеспечивающий как создание пользовательских типов данных, так и расширенные методы запросов, с помощью которых их можно найти. Таким образом, GiST предлагает возможность определить, что вы храните, как вы это храните, а также возможность определять новые пути для поиска через него - возможности которые существенно превышают те, что предлагаются стандартными технологиями (B-дерева, R-дерева и другими).

GiST служит основанием для многих открытых проектов, которые используют PostgreSQL, такие как OpenFTS и PostGIS. OpenFTS (Open Source полнотекстовый поиск), который предоставляет онлайн индексацию данных и актуальности ранжирования для поиска по базе данных. PostGIS это проект, который добавляет поддержку географических объектов в PostgreSQL, что позволяет ему быть использованным в качестве базы пространственных данных для геоинформационных систем (ГИС).

Другие улучшения включают в себя табличное наследование, систему правил, и события баз данных. Табличное наследование придает объектно-ориентированный уклон созданию таблиц, что может быть очень полезно разработчикам баз данных для получения новых таблиц из других таблиц, рассматривая их как базовые классы. Еще лучше, PostgreSQL поддерживает как одинарное, так и множественное наследование в этой манере.

Система правил, также называемая системой перезаписи запросов, позволяет разработчику базы данных создавать правила, которые задают определенные операции для данной таблицы или представления, и динамично трансформируют их в другие операции при их обработке.

PostgreSQL поддерживает хранимые процедуры на более чем десяти языках программирования, в том числе Java, Perl, Python, Ruby, Tcl C / C ++, а также собственный PL / PgSQL, который похож на Oracle, PL / SQL. В стандартную библиотеку функций включены сотни встроенных функций, которые варьируются от базовых математических и строковых операций до операций в криптографии и совместимости с Oracle. Триггеры и хранимые процедуры могут быть написаны на C и загружены в базу данных в качестве библиотеки, что предоставляет большую гибкость в расширении своих возможностей. Также PostgreSQL включает в себя инфраструктуру, которая позволяет разработчикам определять и создавать свои собственные пользовательские типы данных вместе с функциями и операторами, которые определяют их поведение. В результате появилось множество передовых типов данных были созданы, которые варьируются от геометрических и пространственных примитивов сетевых адресов даже ISBN / ISSN (Международный стандартный номер книги / Международный стандартный номер серийного) типы данных, каждый из которых может быть добавлен в систему.

Так же, как существуют много языков процедурных поддерживаемых PostgreSQL, есть и множество интерфейсных библиотек, а также различные языки и обобщены и интерпретированы для взаимодействия с PostgreSQL. Существуют интерфейсы для Java (JDBC), ODBC, Perl, Python, Ruby, C, C ++, PHP, Lisp, Scheme, и Qt просто назвать несколько.

Исходный код в PostgreSQL доступен всем, под либеральной лицензией с открытым исходным кодом. Эта лицензия дает вам свободу использовать, модифицировать и распространять PostgreSQL в любой форме открытой или закрытой. Таким образом, PostgreSQL это не только мощная система базы данных, на которой может работать предприятие, оно представляет собой платформу разработки, на которой можно разрабатывать свои проекты.

3 Практическая часть

3.1 Разработка структуры базы данных

Философия Rails предписывает создания базы данных посредством предоставляемых самим Rails инструментов, без взаимодействия с СУБД напрямую. Для создания таблиц используются миграции строенной в Rails ORM. Это обеспечивает независимость приложения от СУБД, так как ORM обеспечивает разработчика интерфейсами ко всем популярным на сегодняшний день.

Rails также не создает никаких жестких связей в БД, обеспечивая целостность данных на уровне фреймворка.

Структурка БД приложения представлена на рисунке 3.1.

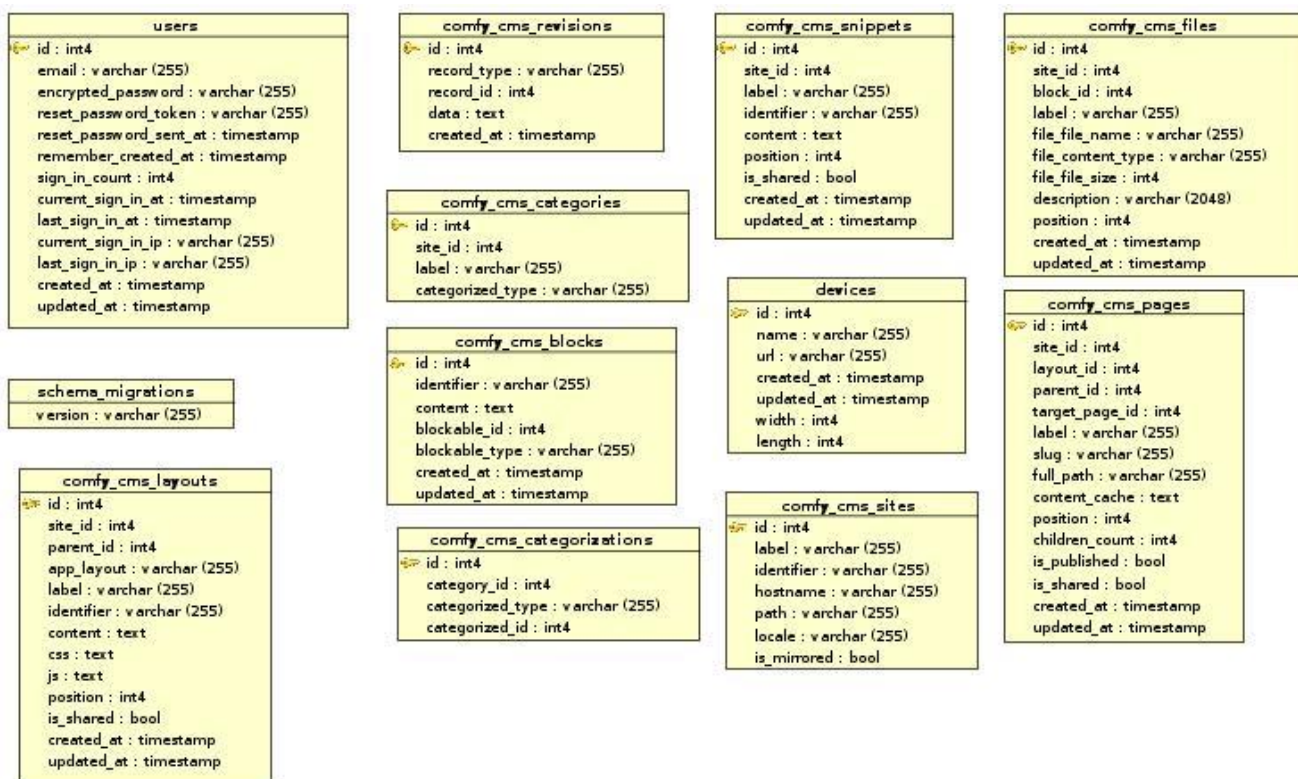


Рисунок 3.1 – Настройка основных параметров сервера LabVIEW

3.2 Настройка сервера виртуальных приборов LabVIEW

Для обоих удаленных сценариев панели (в пределах окружающей среды LabVIEW или встраиваемых в веб-страницы), веб-сервер должен быть настроен и включен на серверной машине.

Три аспекта настройки :

- Каталоги файлов и настройки сети
- Разрешенные клиенты
- Видимый ВП

Процесс настройки происходит следующим образом:

Включение web сервера и заполнение основных настроек: Метки сервера, директории хранения веб файлов, и порт подключения к серверу. Процесс настройки данных параметров представлен на рисунке 3.2.

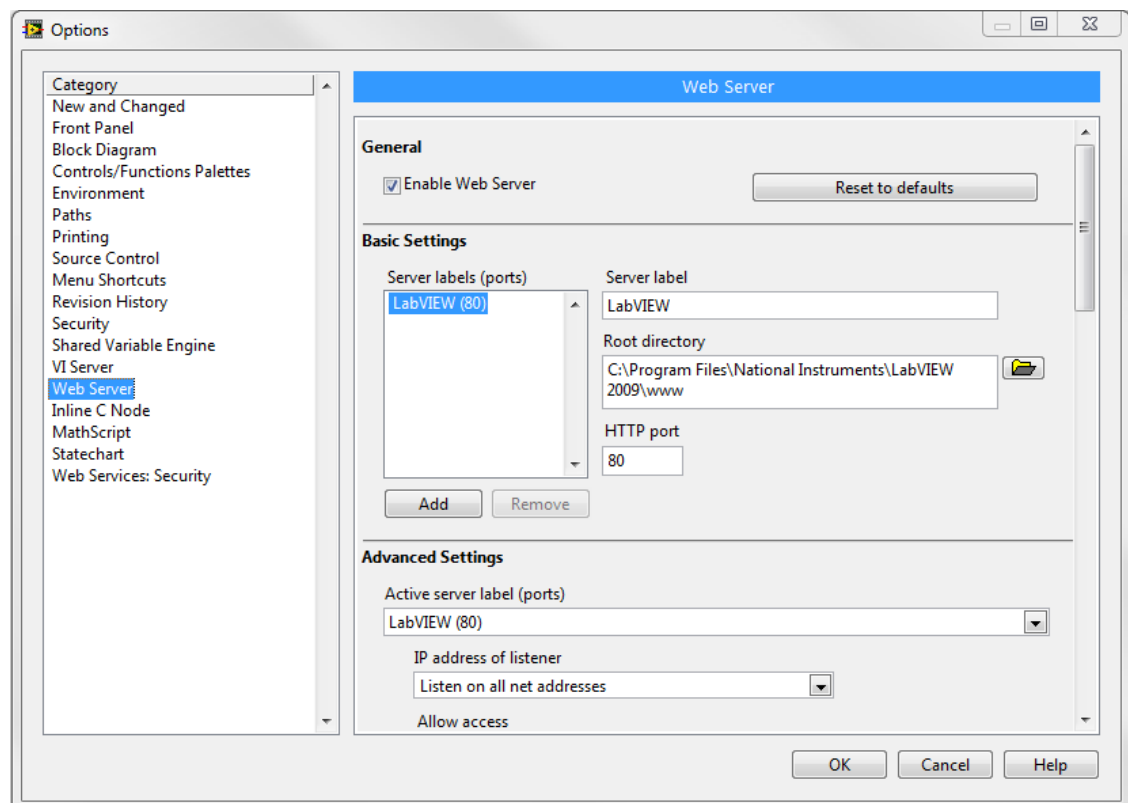


Рисунок 3.2 – Настройка основных параметров сервера LabVIEW

Далее необходимо настроить списки IP адресов имеющих доступ к данному серверу, так называемый Access list, что представлено на рисунке 3.3.

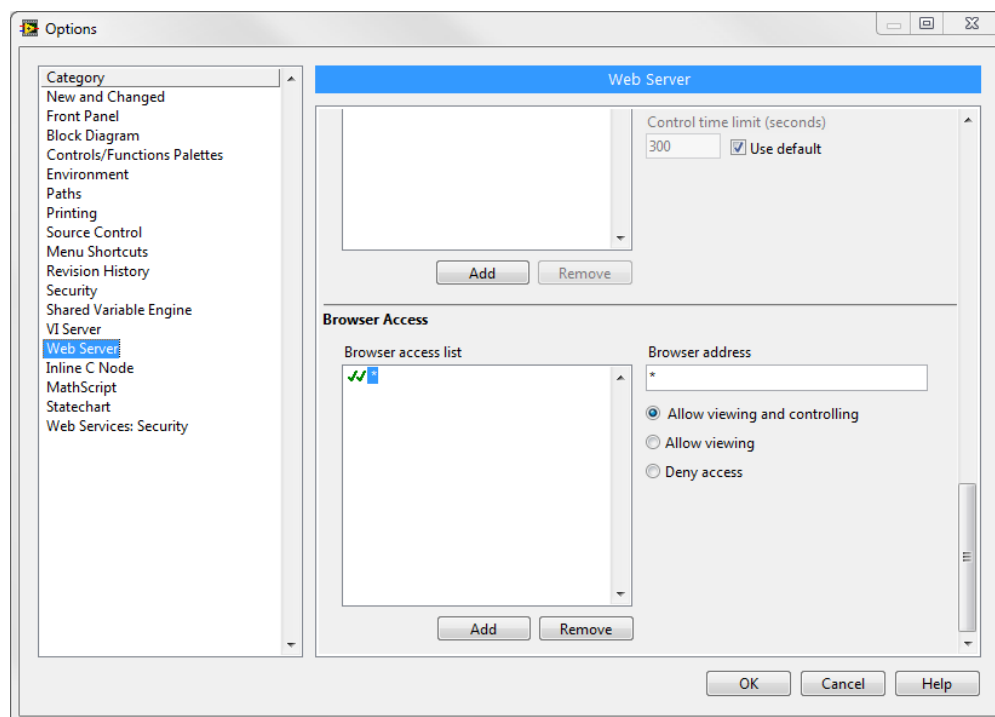


Рисунок 3.3 – Настройка Access list

Заключительным шагом является настройка логирования и определения списка приборов доступных на сервере. Настройка этих параметров изображена на рисунке 3.4.

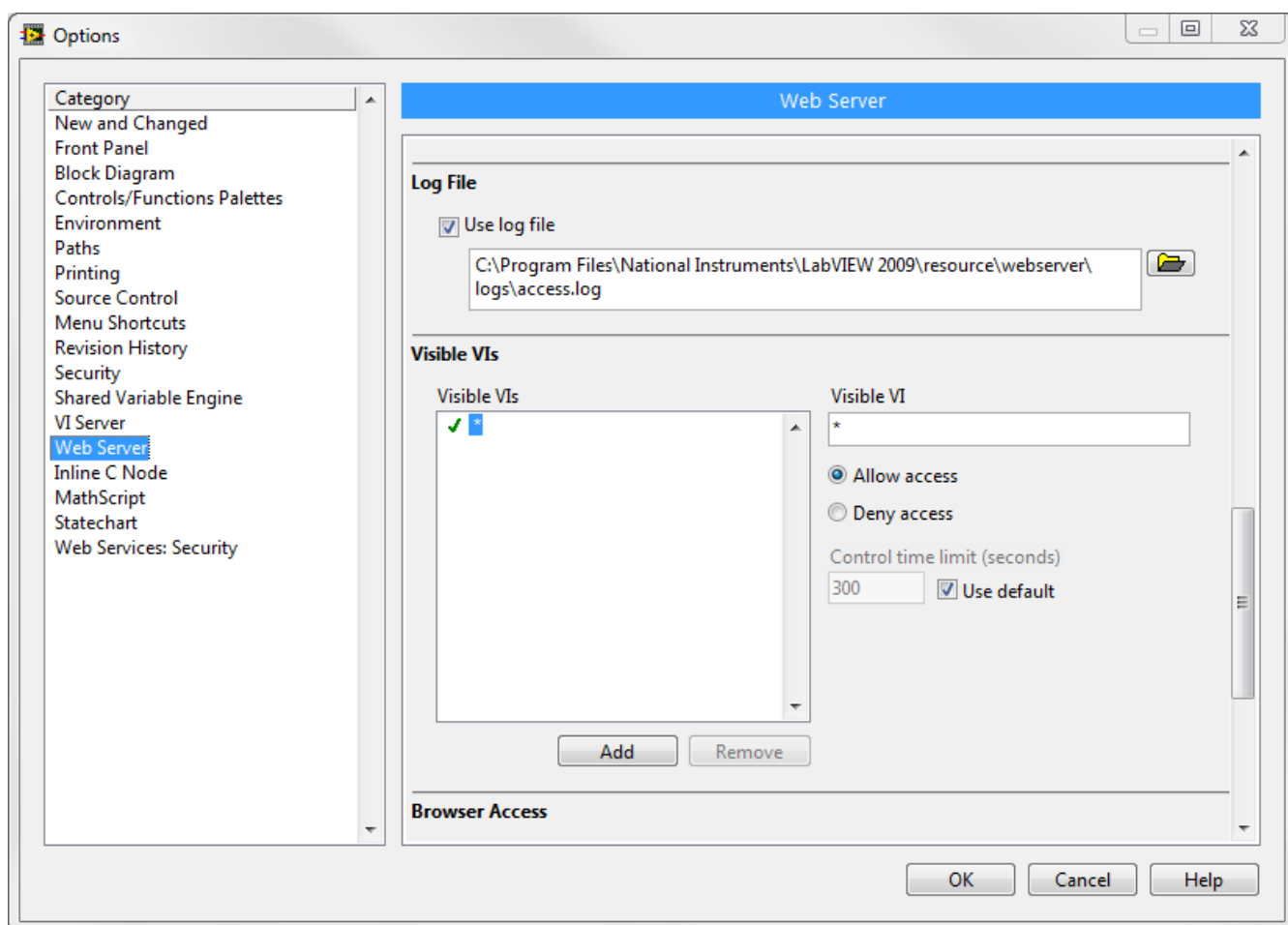


Рисунок 3.4 – Настройка логирования и определения списка приборов доступных на сервере

Заключение

В рамках данной работы были рассмотрены современные технологии удаленного доступа, их применение, в частности в образовании в виде лабораторий удаленного доступа, и методы разработки. Дана оценка актуальности дистанционного обучения с применением таких технологий. Также был проведен анализ некоторых существующих ЛУД.

Углубленно технологии разработки систем удаленного доступа были рассмотрены на примере процесса создания лаборатории удаленного доступа для образовательного учреждения, а именно произведен анализ и выбор технологий разработки системы, методов удаленного соединения между компонентами системы, рассмотрены принципы создания базы данных для такой системы и её структура.

Список использованных источников

- 1 Ferreira, Sousa, Nafalski, Machotka, Nedic (2010). Collaborative learning based on a micro-webserver remote test controller, Bridgeport, University of South Australia, p. 10.
- 2 Обзор мирового опыта создания и эксплуатации лабораторий удаленного доступа. 2011 Постников Е.Б.
- 3 Borisov A.A., Popov N.V., and Shauerman A.A. Foundations of making virtual laboratories in engineering education, International Workshops and Tutorials on Electron Devices and Materials EDM'2006, pp. 180–181, 2006.
- 4 Сайт <http://www.labfor.ru/online>
- 5 Abul K.M. Azad, Michael E. Auer, V. Judson Harward. Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines, 2011.
- 6 Сайт <http://ru.wikipedia.org/>
- 7 Е. А. Роганов, Н. А. Роганова. Программирование на языке Ruby. Учебное пособие (PDF, 425 Кбайт). — М.: МГИУ, 2008. — 56 с. — ISBN 978-5-2760-1495-1
- 8 Брюс Тэйт Практическое использование Rails: Часть 4. Стратегии тестирования в Ruby on Rails. 01.07.2008.
- 9 Хэл Фултон. Программирование на языке Ruby. — 2-е изд. — М.: ДМК Пресс, 2007. — С. 21.
- 10 Thomas, Dave. Extending Ruby (англ.). Programming Ruby — The Pragmatic Programmer's Guide(недоступная ссылка — история). Addison Wesley Longman, Inc.
- 11 Блюм П. LabVIEW: стиль программирования. — М.: ДМК Пресс, 2009.
- 12 Александр Суранов. LabVIEW 8.20. Справочник по функциям. — М.: ДМК Пресс, 2007.

Перечень сокращений

CoC	Convention over Configuration.
DRY	Don't Repeat Yourself.
MVC	Model-View-Controller.
HTML	Hyper Text Markup Language
ВИ	Виртуальный Инструмент.
ВП	Виртуальный Прибор.
СУБД	Система Управления Базами Данных.
ЛУД	Лаборатория Удаленного Доступа.

Приложение А

Листинг web приложения

Представления приборов:

_form.html.erb

```
<%= form_for @device do |f| %>
  <% if @device.errors.any? %>
    <div id="error_explanation">
      <ul>
        <li>
          <%= "#{pluralize(@device.errors.count, "error")} prohibited this device from being saved:" %>
        </li>
      </ul>
      <ul>
        <% @device.errors.full_messages.each do |msg| %>
          <li>
            <%= msg %>
          </li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="row">
    <div class="large-2 columns"><%= f.label :name, class: "right" %></div>
    <div class="large-10 columns"><%= f.text_field :name %></div>
  </div>
  <div class="row">
    <div class="large-2 columns"><%= f.label :url, class: "right" %></div>
    <div class="large-10 columns"><%= f.text_field :url %></div>
  </div>
</div>
```

```

<div class="row">
  <div class="large-2 columns"><%= f.label :width, class: "right" %></div>
  <div class="large-10 columns"><%= f.text_field :width %></div>
</div>

<div class="row">
  <div class="large-2 columns"><%= f.label :length, class: "right" %></div>
  <div class="large-10 columns"><%= f.text_field :length %></div>
</div>

<div class="row actions">
  <div class="large-10 columns large-offset-2">
    <%= f.submit 'Save', class: "button" %>
  </div>
</div>
<% end %>

```

Edit.html.erb

```

<h1>Editing device</h1>
<%= render 'form' %>
<%= link_to 'Show', @device %>
<%= link_to 'Back', devices_path %>

```

Index.html.erb

```

<h1>Listing devices</h1>
<table>
  <tr>
    <th>Name</th>
    <th>URL</th>
    <th></th>
    <th></th>
    <th></th>
  </tr>
  <% @devices.each do |device| %>
    <tr>

```

```

        <td>
            <%= device.name %>
        </td>
        <td>
            <%= device.url %>
        </td>
        <td>
            <%= link_to 'Show', device %>
        </td>
        <td>
            <%= link_to 'Edit', edit_device_path(device) %>
        </td>
        <td>
            <%= link_to 'Destroy', device, :method => :delete, :data => { :confirm => 'Are you sure?' } %>
        </td>
    </tr>
<% end %>
</table>
<br/>
<%= link_to 'New Device', new_device_path, class: "button" %>

```

Show.html.erb

```

<p id="notice">
    <%= notice %>
</p>
<p>
    <b>Name:</b>
    <%= @device.name %>
</p>

<iframe src="<%= @device.url %>" width="<%= @device.length %>"
    height="<%= @device.width %>" seamless="seamless">
    <p>Your browser does not support iframes.</p>
</iframe>

```



```
<%= link_to 'Edit', edit_device_path(@device) %>
<%= link_to 'Back', devices_path %>
```

Макет приложения:

Application.html.erb

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title><%= content_for?(:title) ? yield(:title) : "Remote Lab" %></title>
    <%= stylesheet_link_tag "application", "data-turbolinks-track" => true %>
    <%= javascript_include_tag "vendor/modernizr" %>
    <%= csrf_meta_tags %>
  </head>
  <body>
    <div class='header'>
      <%= render "shared/menu" %>
    </div>
    <div class='content'>
      <div class="row">
        <div class="small-12 columns">
          <%= yield %>
        </div>
      </div>
    </div>
    <div class='footer'>
    </div>
    <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
  </body>
</html>
```

Файлы конфигурации приложения:

Development.rb

```
RemoteLab::Application.configure do
  # Settings specified here will take precedence over those in config/application.rb.

  # In the development environment your application's code is reloaded on
  # every request. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.cache_classes = false

  # Do not eager load code on boot.
  config.eager_load = false

  # Show full error reports and disable caching.
  config.consider_all_requests_local = true
  config.action_controller.perform_caching = false

  # devise mailer
  config.action_mailer.default_url_options = { host: 'localhost:3000' }

  # Don't care if the mailer can't send.
  config.action_mailer.raise_delivery_errors = false

  # Print deprecation notices to the Rails logger.
  config.active_support.deprecation = :log

  # Raise an error on page load if there are pending migrations
  config.active_record.migration_error = :page_load

  # Debug mode disables concatenation and preprocessing of assets.
  # This option may cause significant delays in view rendering with a large
  # number of complex assets.
```

```
config.assets.debug = true
```

```
end
```

production.rb

```
RemoteLab::Application.configure do
```

```
  # Settings specified here will take precedence over those in config/application.rb.
```

```
  # Code is not reloaded between requests.
```

```
  config.cache_classes = true
```

```
  # Eager load code on boot. This eager loads most of Rails and
```

```
  # your application in memory, allowing both thread web servers
```

```
  # and those relying on copy on write to perform better.
```

```
  # Rake tasks automatically ignore this option for performance.
```

```
  config.eager_load = true
```

```
  # Full error reports are disabled and caching is turned on.
```

```
  config.consider_all_requests_local = false
```

```
  config.action_controller.perform_caching = true
```

```
  # Enable Rack::Cache to put a simple HTTP cache in front of your application
```

```
  # Add `rack-cache` to your Gemfile before enabling this.
```

```
  # For large-scale production use, consider using a caching reverse proxy like nginx, varnish or squid.
```

```
  # config.action_dispatch.rack_cache = true
```

```
  # Disable Rails's static asset server (Apache or nginx will already do this).
```

```
  config.serve_static_assets = false
```

```
  # Compress JavaScripts and CSS.
```

```
  config.assets.js_compressor = :uglifier
```

```
  # config.assets.css_compressor = :sass
```

```
  # Do not fallback to assets pipeline if a precompiled asset is missed.
```

```
config.assets.compile = false

# Generate digests for assets URLs.
config.assets.digest = true

# Version of your assets, change this if you want to expire all your assets.
config.assets.version = '1.0'

# Specifies the header that your server uses for sending files.
# config.action_dispatch.x_sendfile_header = "X-Sendfile" # for apache
# config.action_dispatch.x_sendfile_header = 'X-Accel-Redirect' # for nginx

# Force all access to the app over SSL, use Strict-Transport-Security, and use secure cookies.
# config.force_ssl = true

# Set to :debug to see everything in the log.
config.log_level = :info

# Prepend all log lines with the following tags.
# config.log_tags = [ :subdomain, :uuid ]

# Use a different logger for distributed setups.
# config.logger = ActiveSupport::TaggedLogging.new(SyslogLogger.new)

# Use a different cache store in production.
# config.cache_store = :mem_cache_store

# Enable serving of images, stylesheets, and JavaScripts from an asset server.
# config.action_controller.asset_host = "http://assets.example.com"

# Precompile additional assets.
# application.js, application.css, and all non-JS/CSS in app/assets folder are already added.
# config.assets.precompile += %w( search.js )
```

```

# Ignore bad email addresses and do not raise email delivery errors.
# Set this to true and configure the email server for immediate delivery to raise delivery errors.
# config.action_mailer.raise_delivery_errors = false

# Enable locale fallbacks for I18n (makes lookups for any locale fall back to
# the I18n.default_locale when a translation can not be found).
config.i18n.fallbacks = true

# Send deprecation notices to registered listeners.
config.active_support.deprecation = :notify

# Disable automatic flushing of the log to improve performance.
# config.autoflush_log = false

# Use default logging formatter so that PID and timestamp are not suppressed.
config.log_formatter = ::Logger::Formatter.new
end

```

application.rb

```

require File.expand_path('../boot', __FILE__)

require 'rails/all'

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(:default, Rails.env)

module RemoteLab
  class Application < Rails::Application
    # Settings in config/environments/* take precedence over those specified here.
    # Application configuration should go into files in config/initializers
    # -- all .rb files in that directory are automatically loaded.

```

```

# Set Time.zone default to the specified zone and make Active Record auto-convert to this zone.
# Run "rake -D time" for a list of tasks for finding time zone names. Default is UTC.
# config.time_zone = 'Central Time (US & Canada)'

# The default locale is :en and all translations from config/locales/*.rb,yml are auto loaded.
# config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}').to_s]
# config.i18n.default_locale = :de
end
end

```

Comfortable_mexican_sofa.rb

```

# encoding: utf-8

ComfortableMexicanSofa.configure do |config|
  # Title of the admin area
  # config.cms_title = 'ComfortableMexicanSofa CMS Engine'

  # Controller that is inherited from CmsAdmin::BaseController
  # config.base_controller = 'ApplicationController'

  # Module responsible for authentication. You can replace it with your own.
  # It simply needs to have #authenticate method. See http_auth.rb for reference.
  # config.admin_auth = 'ComfortableMexicanSofa::HttpAuth'

  # Module responsible for public authentication. Similar to the above. You also
  # will have access to @cms_site, @cms_layout, @cms_page so you can use them in
  # your logic. Default module doesn't do anything.
  # config.public_auth = 'ComfortableMexicanSofa::DummyAuth'

  # When arriving at /cms-admin you may chose to redirect to arbitrary path,
  # for example '/cms-admin/users'
  # config.admin_route_redirect = "

  # File uploads use Paperclip and can support filesystem or s3 uploads. Override

```

```

# the upload method and appropriate settings based on Paperclip. For S3 see:
# http://rdoc.info/gems/paperclip/2.3.8/Paperclip/Storage/S3, and for
# filesystem see: http://rdoc.info/gems/paperclip/2.3.8/Paperclip/Storage/Filesystem
# If you are using S3 and HTTPS, pass :s3_protocol => " to have URLs that use the protocol of the
page
# config.upload_file_options = { :url => '/system/:class/:id/:attachment/:style/:filename' }

# Sofa allows you to setup entire site from files. Database is updated with each
# request (if necessary). Please note that database entries are destroyed if there's
# no corresponding file. Fixtures are disabled by default.
# config.enable_fixtures = false

# Path where fixtures can be located.
# config.fixtures_path = File.expand_path('db/cms_fixtures', Rails.root)

# Importing fixtures into Database
# To load fixtures into the database just run this rake task:
# local: $ rake comfortable_mexican_sofa:fixtures:import FROM=example.local TO=localhost
# Heroku: $ heroku run rake comfortable_mexican_sofa:fixtures:import FROM=example.local
TO=yourapp.herokuapp.com
# From indicates folder the fixtures are in and to is the Site hostname you have defined in the
database.

# Exporting fixtures into Files
# If you need to dump database contents into fixture files run:
# local: $ rake comfortable_mexican_sofa:fixtures:export FROM=localhost TO=example.local
# Heroku: $ heroku run rake comfortable_mexican_sofa:fixtures:export
FROM=yourapp.herokuapp.com TO=example.local
# This will create example.local folder and dump all content from example.com Site.

# Content for Layouts, Pages and Snippets has a revision history. You can revert
# a previous version using this system. You can control how many revisions per
# object you want to keep. Set it to 0 if you wish to turn this feature off.
config.revisions_limit = 25

```

```
# Locale definitions. If you want to define your own locale merge
# {:locale => 'Locale Title'} with this.
config.locales = {:en => 'English', :es => 'Español'}

# Admin interface will respect the locale of the site being managed. However you can
# force it to English by setting this to `:en`
config.admin_locale = nil

# A class that is included as a sweeper to admin base controller if it's set
# config.admin_cache_sweeper = nil

# By default you cannot have irb code inside your layouts/pages/snippets.
# Generally this is to prevent putting something like this:
# <% User.delete_all %> but if you really want to allow it...
config.allow_irb = false

# Whitelist of all helper methods that can be used via {{cms:helper}} tag. By default
# all helpers are allowed except `eval`, `send`, `call` and few others. Empty array
# will prevent rendering of all helpers.
# config.allowed_helpers = nil

# Whitelist of partials paths that can be used via {{cms:partial}} tag. All partials
# are accessible by default. Empty array will prevent rendering of all partials.
config.allowed_partials = nil

# Site aliases, if you want to have aliases for your site. Good for harmonizing
# production env with dev/testing envs.
# e.g. config.hostname_aliases = {'host.com' => 'host.inv', 'host_a.com' => ['host.lvh.me', 'host.dev']}
# Default is nil (not used)
config.hostname_aliases = nil

# Reveal partials that can be overwritten in the admin area.
```



```

# Default is false.
# config.reveal_cms_partials = false

end

# Default credentials for ComfortableMexicanSofa::HttpAuth
# YOU REALLY WANT TO CHANGE THIS BEFORE PUTTING YOUR SITE LIVE
ComfortableMexicanSofa::HttpAuth.username = 'admin'
ComfortableMexicanSofa::HttpAuth.password = '123456'

# You can use bcrypt (gem 'bcrypt-ruby') if you want to:
require 'bcrypt'
ComfortableMexicanSofa::HttpAuth.username = 'username'
ComfortableMexicanSofa::HttpAuth.password = BCrypt::Password.new '... bcrypt hash ...'
#
# To create a bcrypt hash:
# BCrypt::Password.create('password').to_s

```

Основные миграции:

Создание таблиц CMS

```

class CreateCms < ActiveRecord::Migration

  def self.up

    text_limit = case ActiveRecord::Base.connection.adapter_name
    when 'PostgreSQL'
      { }
    else
      { :limit => 16777215 }
    end

    # -- Sites -----
    create_table :comfy_cms_sites do |t|

```

```

t.string :label,      :null => false
t.string :identifier, :null => false
t.string :hostname,   :null => false
t.string :path
t.string :locale,     :null => false, :default => 'en'
t.boolean :is_mirrored, :null => false, :default => false
end

add_index :comfy_cms_sites, :hostname
add_index :comfy_cms_sites, :is_mirrored

# -- Layouts -----
create_table :comfy_cms_layouts do |t|
  t.integer :site_id,      :null => false
  t.integer :parent_id
  t.string :app_layout
  t.string :label,         :null => false
  t.string :identifier, :null => false
  t.text :content,        text_limit
  t.text :css,            text_limit
  t.text :js,             text_limit
  t.integer :position,     :null => false, :default => 0
  t.boolean :is_shared,   :null => false, :default => false
  t.timestamps
end

add_index :comfy_cms_layouts, [:parent_id, :position]
add_index :comfy_cms_layouts, [:site_id, :identifier], :unique => true

# -- Pages -----
create_table :comfy_cms_pages do |t|
  t.integer :site_id,      :null => false
  t.integer :layout_id
  t.integer :parent_id
  t.integer :target_page_id

```

```

t.string :label, :null => false
t.string :slug
t.string :full_path, :null => false
t.text :content_cache, text_limit
t.integer :position, :null => false, :default => 0
t.integer :children_count, :null => false, :default => 0
t.boolean :is_published, :null => false, :default => true
t.boolean :is_shared, :null => false, :default => false
t.timestamps
end

add_index :comfy_cms_pages, [:site_id, :full_path]
add_index :comfy_cms_pages, [:parent_id, :position]

# -- Page Blocks -----
create_table :comfy_cms_blocks do |t|
  t.string :identifier, :null => false
  t.text :content, text_limit
  t.references :blockable, :polymorphic => true
  t.timestamps
end

add_index :comfy_cms_blocks, [:identifier]
add_index :comfy_cms_blocks, [:blockable_id, :blockable_type]

# -- Snippets -----
create_table :comfy_cms_snippets do |t|
  t.integer :site_id, :null => false
  t.string :label, :null => false
  t.string :identifier, :null => false
  t.text :content, text_limit
  t.integer :position, :null => false, :default => 0
  t.boolean :is_shared, :null => false, :default => false
  t.timestamps
end

```

```

add_index :comfy_cms_snippets, [:site_id, :identifier], :unique => true
add_index :comfy_cms_snippets, [:site_id, :position]

# -- Files -----
create_table :comfy_cms_files do |t|
  t.integer :site_id,      :null => false
  t.integer :block_id
  t.string  :label,        :null => false
  t.string  :file_file_name, :null => false
  t.string  :file_content_type, :null => false
  t.integer :file_file_size, :null => false
  t.string  :description,   :limit => 2048
  t.integer :position,     :null => false, :default => 0
  t.timestamps
end
add_index :comfy_cms_files, [:site_id, :label]
add_index :comfy_cms_files, [:site_id, :file_file_name]
add_index :comfy_cms_files, [:site_id, :position]
add_index :comfy_cms_files, [:site_id, :block_id]

# -- Revisions -----
create_table :comfy_cms_revisions, :force => true do |t|
  t.string  :record_type, :null => false
  t.integer :record_id,   :null => false
  t.text    :data,        text_limit
  t.datetime :created_at
end
add_index :comfy_cms_revisions, [:record_type, :record_id, :created_at],
  :name => 'index_cms_revisions_on_rtype_and_rid_and_created_at'

# -- Categories -----
create_table :comfy_cms_categories, :force => true do |t|
  t.integer :site_id,      :null => false

```

```

    t.string :label, :null => false
    t.string :categorized_type, :null => false
  end
  add_index :comfy cms_categories, [:site_id, :categorized_type, :label], :unique => true,
    :name => 'index cms_categories_on_site_id_and_cat_type_and_label'

  create_table :comfy cms_categorizations, :force => true do |t|
    t.integer :category_id, :null => false
    t.string :categorized_type, :null => false
    t.integer :categorized_id, :null => false
  end
  add_index :comfy cms_categorizations, [:category_id, :categorized_type, :categorized_id], :unique
=> true,
    :name => 'index cms_categorizations_on_cat_id_and_catd_type_and_catd_id'
end

def self.down
  drop_table :comfy cms_sites
  drop_table :comfy cms_layouts
  drop_table :comfy cms_pages
  drop_table :comfy cms_snippets
  drop_table :comfy cms_blocks
  drop_table :comfy cms_files
  drop_table :comfy cms_revisions
  drop_table :comfy cms_categories
  drop_table :comfy cms_categorizations
end
end

```

Создания таблиц Devise

```

class DeviseCreateUsers < ActiveRecord::Migration
  def change
    create_table(:users) do |t|
      ## Database authenticatable

```

```

t.string :email,          null: false, default: ""
t.string :encrypted_password, null: false, default: ""

## Recoverable
t.string :reset_password_token
t.datetime :reset_password_sent_at

## Rememberable
t.datetime :remember_created_at

## Trackable
t.integer :sign_in_count, default: 0, null: false
t.datetime :current_sign_in_at
t.datetime :last_sign_in_at
t.string :current_sign_in_ip
t.string :last_sign_in_ip

## Confirmable
# t.string :confirmation_token
# t.datetime :confirmed_at
# t.datetime :confirmation_sent_at
# t.string :unconfirmed_email # Only if using reconfirmable

## Lockable
# t.integer :failed_attempts, default: 0, null: false # Only if lock strategy is :failed_attempts
# t.string :unlock_token # Only if unlock strategy is :email or :both
# t.datetime :locked_at

t.timestamps
end

add_index :users, :email,          unique: true

```

```
add_index :users, :reset_password_token, unique: true
# add_index :users, :confirmation_token, unique: true
# add_index :users, :unlock_token, unique: true
end
end
```

Создание некоторых пользовательских таблиц

```
class CreateDevices < ActiveRecord::Migration
  def change
    create_table :devices do |t|
      t.string :name
      t.string :url

      t.timestamps
    end
  end
end

class AddSizesToDevices < ActiveRecord::Migration
  def change
    add_column :devices, :width, :integer
    add_column :devices, :length, :integer
  end
end
```