# Министерство образования и науки Российской Федерации

# Федеральное государственное бюджетное образовательное учреждение

# высшего профессионального образования

# «Ижевский государственный технический университет имени М. Т. Калашникова»

# Кафедра «Программное обеспечение»

Отчет по лабораторной работе №2

по дисциплине: «Конструирование ПО»

на тему: «Умный дом»

Выполнили

студенты гр. Б08-191-1

В.Д. Шутов

М.И. Копылов

А.А. Найдин

Принял В.Г. Власов

Ижевск

2016

1. ВВЕДЕНИЕ

1.1 Назначение системы

Данная система необходима для обеспечения безопасности, а также дистанционного контроля и автоматизации работы основных домашних систем.

1.2 Область применения системы

Система «Умный дом» применяется в загородных домах и городских квартирах для обеспечения безопасности и автоматизации работы основных бытовых систем. Данная система представляет собой совокупность устройств, считывающих различную информацию, устройств, осуществляющих определенные действия, контроллера, принимающего данные с устройств и сервера, обрабатывающего данные, а так же административной панели, в которой осуществляется контроль над всеми установленными системами «Умный дом».

Основные функциональные возможности системы:

1. Сбор данных с устройств
2. Сохранение определенных данных в БД
3. СМС уведомление о пожарной тревоге или срабатывании охранной сигнализации
4. Вывод на экран пользователя данных с устройств
5. Сбор статистических данных
6. Отображение графиков

1.5 Обзор системы

Система делится на четыре главных модуля: Контроллер, Обработчик, Сервер, Пользовательский интерфейс.

Контроллер собирает показания устройств с заданной периодичностью, а обработчик принимает решение в соответствии с настройками, включает, настраивает другие устройства, отправляет данные в БД, принимает настройки от сервера.

Сервер получает данные от БУ, отправляет настройки на БУ, сохраняет данные в БД. Так же на сервере находится административная панель, в которой осуществляется обработка информации с каждой из установленных систем, а так же меняются их настройки.

Пользовательский интерфейс взаимодействует с обработчиком, из которого получает необходимые данные и, в зависимости от команды пользователя, обрабатывает эти данные и отправляет их на контроллер и в БД.

2. ОБЩЕЕ ОПИСАНИЕ СИСТЕМЫ

2.1 Системный контекст

2.2 Режимы и состояния системы

Система может находиться в следующих состояниях:

1. Состояния контроллера.

* ожидание опроса устройства;
* опрос устройств;
* проверка соответствия показаний устройств настройкам;
* включение исполнительных устройств.

1. Состояния обработчика.

* ожидание приема данных;
* прием данных;
* сохранение данных в БД;
* загрузка данных из БД;
* отправка данных в контроллер

1. Состояния интерфейса.

* загрузка данных из обработчика;
* отправления настроек в обработчик;
* ожидание команды пользователя;
* ввод команды пользователем.

1. Состояние сервера

* Ожидание авторизации пользователем
* Авторизация пользователя
* Переадресация пользователя на обработчик
* Авторизация в панели администратора
* Изменение настроек систем
* Получение данных от систем

2.3 Основные функциональные возможности системы



2.4 Основные условия системы

Сервер должен иметь постоянный доступ к интернету.

Внешний IP системы должен быть статическим

2.5 Основные ограничения системы

Начальная версия системы будет работать только на территории УР, г. Ижевск

2.6 Характеристики пользователя

1. Пользователь. Конечный пользователь системы. Количество пользователей этого типа не ограничено. Используют систему для контроля над своим домом.
2. Администратор. Имеет возможность наблюдать за любой установленной системой и менять настройки. Пользователей этого типа несколько, в зависимости от количества установленных систем. Изначально пользователь такого типа один.

2.7 Допущения и зависимости

2.8 Оперативные сценарии

Система предусматривает следующий сценарий. Контроллеропрашивает устройство и отправляет данные на обработчик. Обработчик, принимает решение о необходимости включить соответствующие исполнительные устройства. Также он отправляет данные в БД и принимает от сервера новые настройки. Обработчик принимает данные и сохраняет их в БД. Если в БД были записаны новые настройки, обработчик их отправляет в контроллер. Пользователь запускает web-приложение(интерфейс) и вводит необходимые команды, а так же наблюдает за работой системы и её текущим состоянием. Администратор авторизуется в панели администратора и может устанавливать новые настройки у любой системы.

ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ, УСЛОВИЯ И ОГРАНИЧЕНИЯ СИСТЕМЫ

* 1. Физические
     1. Конструкция

Контроллер и обработчик состоят из одноплатного компьютера Raspberry Pi 2и подключенных к нему устройств. Обработчик подключается к серверу с помощью сетевого кабеля через роутер.

3.1.2 Износостойкость

Данный параметр системы не зависит от нас, т.к. сервер арендован.

3.1.3 Адаптируемость

В случае увеличения нагрузки на сервер, количество арендованных серверов будет увеличено, а нагрузка равномерно распределена между ними.

3.1.4 Условия окружающей среды

3.2. Рабочие характеристики системы

Критическое количество запросов, обрабатываемое системой зависит от установленного одноплатного компьютера. Долговечность оборудование зависит от поставщика сервера. До превышения критического количества запросов в час пользователь не замечает нагрузку оборудования.

3.3 Защита системы

Система должна быть защищена от sql - инъекций и xss - атак.

3.4 Информационный менеджмент

3.5 Работа системы

3.5.1 Эргономика системы

Требования по распределению функций для персонала описано в ТЗ

3.5.2 Эксплуатационная технологичность

Техническая составляющая эксплуатационной технологичности зависит от арендодателя сервера. Программные изменения происходят в фоновом режиме и не вызывают задержек при обработке команд.

3.5.3 Надежность системы

Техническая составляющая надежности системы зависит от арендодателя сервера.

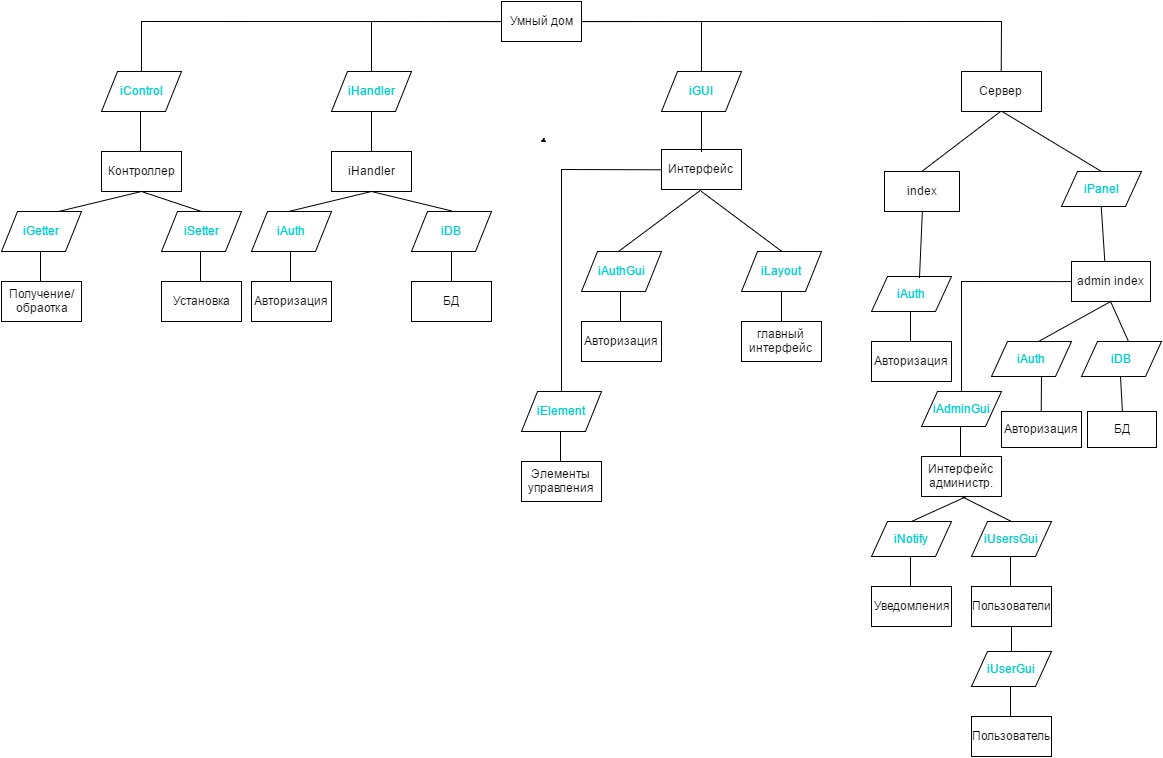
3.6 Стратегия и регулирование

Базовые настройки установленной системы регулируются при установке системы администратором, в зависимости от пожеланий пользователя.

3.7 Устойчивость жизненного цикла системы

Для устойчивости жизненного цикла системы используется обратная связь с пользователем.

4. СИСТЕМНЫЕИНТЕРФЕЙСЫ



Интерфейсы Admin Panel:

1. Интерфейс модуляAdmin

interface iAdmin

{

function ShowAuth();

function ShowGui();

function InitDB($param);

function ShowUsers();

}

2. Интерфейс модуляAdmin Gui

interface iAdminGui

{

function Show();

function GetUsers();

function GetUser();

}

3. Интерфейс модуля Авторизации

interface iAuth

{

function CheckAuth();

function MakeAuth();

}

4. Интерфейс модуля работысБД

interface iDB

{

function Connect($param);

function Query($query);

function Die(\Exception $e);

}

5. Интерфейс модуля Пользователь

interface iUser

{

function SendNote();

function AddModule($module);

function DeleteModule($module);

}

6. Интерфейс модуля iUserGui

interface iUserGui

{

static function GetUserInfo(User $user);

function EditUser(User $user);

}

7. Интерфейс модуля iUsersGui

interfaceiUsersGui

{

function GetUsersList();

function SearchUser($param);

}

Интерфейсы основной системы

1. Интерфейс модуля iAlarm

interface iAlarm

{

public function sendSms();

public function callToPolice();

public function callTo112();

}

1. Интерфейс модуля iAuth

interface iAuth

{

public function Validation();

public function getAuth();

public function logOut();

public function isAuth($username, DB $db = null);

}

1. Интерфейс модуля iController

interface iController

{

public function makeInteraction(Handler $handler);

public function makeAlarm(Alarm $alarm);

}

1. Интерфейс модуля iDB

interface iDB

{

public function connect();

public function authorization();

}

1. Интерфейс модуля iHandler

interface iHandler

{

public function makeAuth(Auth $auth = null);

public function makeDB(DB $db = null);

public function makeInterController();

}

1. Интерфейс модуля iInterController

interface iInterController

{

public function getDataByTime();

public function getDataById($id);

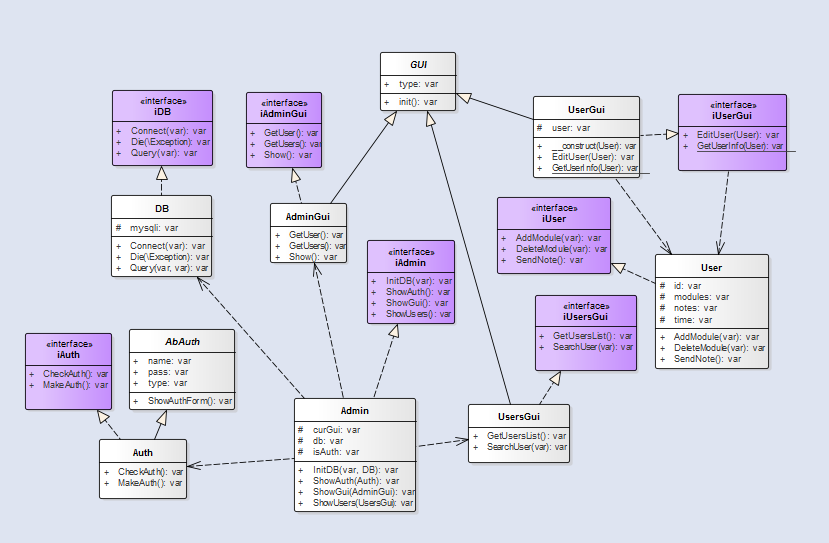
public function sendData(DB $db = null, $id, $value);

}

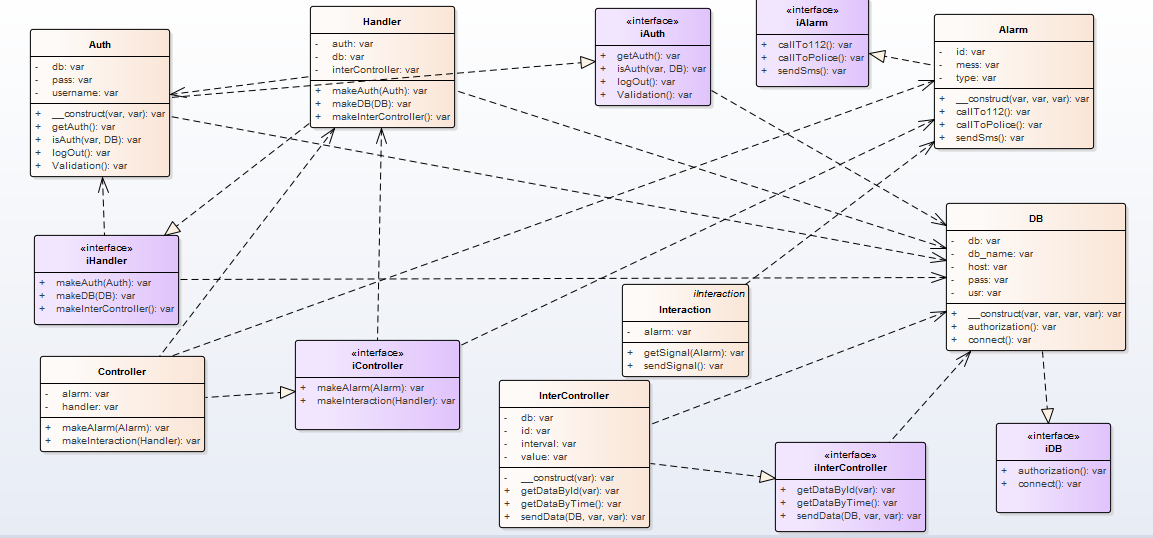
1. КЛАССЫКОДИРОВАНИЯ

5.1 UML схема программы

AdminPanel



General system



Классы кодирования

5.2 Классы кодирования программы соответствуют по функциональному назначению интерфейсам, по которым построены классы кодирования (см. п.4 Системные интерфейсы). Для системы так же разработаны следующие классы представления данных:

1. Класс Admin

class Admin implements iAdmin

{

protected $isAuth, $curGui, $db;

}

2. Класс DB

class DB implements iDB

{

protected $mysqli;

}

3. Класс GUI

abstract class GUI

{

public $type;

public function init() {

require\_once("interface/".$this->type.".html");

}

}

4. Класс User

class User implements iUser

{

protected $id, $modules, $time, $notes;

5. Класс UserGui

class UserGui extends \GUI implements iUserGui

{

protected $user;

}

6. Класс Alarm

class Alarm implements iAlarm

{

private $id;

private $mess;

private $type;

}

7. Класс Auth

class Auth implements iAuth

{

private $username;

private $pass;

private $db;

}

8. Класс Controller

class Controller implements iController

{

private $handler;

private $alarm;

}

9. Класс DB

class DB implements iDB

{

private $db;

private $host;

private $pass;

private $usr;

private $db\_name;

}

10. Класс Handler

class Handler implements iHandler

{

private $auth;

private $db;

private $interController;

}

11. Класс Interaction

class Interaction implements iInteraction

{

private $alarm;

}

12. Класс InterController

class InterController implements iInterController

{

private $id;

private $value;

private $interval;

private $db;

}