

MagnisTrade.com

Scripte php

Requisitos para o funcionamento do script:

Não tem como garantir o funcionamento fora desses requisitos abaixo:

- Servidor Linux com cPanel da (cPanel.net)
- PHP 5.4 a 5.5
- MySQL
- Apache
- phpMyAdmin

Acesse o cPanel

Versão do PHP

Verifique se a versão do PHP que está usando corresponde com as versões informadas acima, caso contrário, altere a versão do PHP, geralmente essa opção no cPanel fica em “Selecionar Versão do PHP”.

Configuração do banco de dados

Agora vá ao assistente de criação de banco de dados MySQL. Crie o Banco de Dados MySQL, o Usuário de acesso ao banco + senha, e atribua todas as permissões do usuário ao Banco.

Importar a base de dados que está dentro da pasta /INSTALACAO

- Abra o phpMyAdmin selecione o banco que criou, vá em “IMPORTAR”.
- Importe a base de dados **BASE-DE-DADOS.sql** que está dentro da pasta /INSTALACAO

Edição do arquivo que conecta com o banco de dados:

Nos arquivos do script que estão dentro da pasta /script, acesse o seguinte arquivo e edite conforme os dados do banco mysql que criou anteriormente.

- Abrir o arquivo /application/config/database.php

Alterar os dados conforme seu servidor (linhas 67 a 70):

```
'hostname' => 'localhost', . <- Aqui o host de conexão MySQL. Geralmente é localhost mesmo  
'username' => 'USUARIOCPANEL_USUARIOBANCO', <- Aqui o usuario do banco.  
'password' => 'SENHA', <- aqui a senha  
'database' => 'USUARIOCPANEL_BANCO', <- Aqui o banco.
```

- Salve o arquivo.

Enviando os Arquivos para o servidor:

- Agora abra a pasta /script

- Compacte o conteúdo da pasta /Script em um arquivo .zip (inclusive o .htaccess).

- Abra o cPanel, vá em Gerenciador de Arquivos, abra a pasta que irá jogar os arquivos, exemplo:

/public_html para instalação direta na raiz do domínio, ex: seusite.com

/public_html/pasta/ - para instalação em uma pasta, ex: seusite.com/pasta

- Ainda no Gerenciador de Arquivos vá em “Carregar” ou “Upload”, e selecione o arquivo zipado.

Faça o upload deste arquivo zipado pelo Gerenciador de Arquivos do cPanel e assim que finalizar o upload volte na pasta que jogou o arquivo zipado, selecione esse arquivo e vá em “Extrair”, então tudo será extraído do arquivo zipado.

- Pronto.

Acesso a área Administrativa:

Acesse www.seusite.com.br/admin ou www.seusite.com.br/PASTA/admin

Email padrão: admin@admin.com

Senha padrão: admin

Altere a senha após acessar o painel do sistema!

Estrutura do código

A aplicação é desenvolvida em Codeigniter framework e segue o padrão MVC completamente.

O diretório root contém

- Application

contém os principais arquivos do aplicativo

- Assets

Conteúms todos os arquivos css e javascript

- system

contém todas as configurações e arquivos de biblioteca da estrutura

- uploads

 - admin_image

 - contém a imagem do perfil carregada pelo próprio administrador

- doctor_image

contém imagens carregadas de médicos - patient_image

contém imagens carregadas de pacientes - nurse_image

contém as imagens carregadas de enfermeiros - farmácia_image

contém as imagens carregadas de farmacêuticos - laboratorist_image

contém as imagens carregadas de laboratoristas - accountant_image

contém as imagens carregadas de contadores - receptionist_image

contém as imagens carregadas de recepcionistas - diagnóstico_report

contém os documentos carregados dos relatórios de diagnóstico - index.php

Este arquivo no carregamento chama todo o aplicativo de funções do codeigniter

- config

contém os arquivos de configuração do aplicativo

- controllers

 - admin.php

 - doctor.php

 - patient.php

 - nurse.php

 - pharmacist.php

 - laboratorist.php

 - accountant.php

 - receptionist.php

 - error.php

 - login.php

 - modal.php

 - payment.php

- helpers

- libraries

- models

 - crud_model.php

 - email_model.php

 - sms_model.php

- views

 - backend

 - admin

contém todos os arquivos de exibição do painel de administração - médico

contém todos os arquivos de exibição do painel de médicos - paciente

contém todos os arquivos de exibição do painel do paciente - enfermeira

contém todos os arquivos de exibição do painel de enfermagem - farmacêutico

contém todos os arquivos de exibição do painel de farmacêutico - laboratorista

contém todos os arquivos de exibição do painel laboratorista - contador

contém todos os arquivos de exibição do painel contador - recepcionista

contém todos os arquivos de exibição do painel do recepcionista

 - footer.php

 - header.php

 - includes_bottom.php

contém links para arquivos javascript - includes_top.php

contém links para arquivos css

 - index.html

 - index.php

 - login.php

 - A página de login

 - modal.php

Descrição do código-fonte

Doctor Controller (/application/controllers/doctor.php)

Gerenciar compromissos

A função de compromisso contém as expressões lógicas de mostrar todos os compromissos criados pelo médico registrado, criando um novo compromisso, editando as informações de um compromisso existente e excluindo as informações da consulta do banco de dados. Os parâmetros definem a ação do formulário em que os dados devem ser criados, editados ou excluídos. Um sms é enviado ao paciente quando um novo compromisso é criado ou editado chamando função send_sms com mensagem e número de telefone do receptor como parâmetros definidos em / application / models / sms_model.php

```

function save_appointment_info()
{
    $data['timestamp'] = strtotime($this->input->post('date_timestamp'));
    $data['status'] = 'approved';
    $data['patient_id'] = $this->input->post('patient_id');

    if($this->session->userdata('login_type') == 'doctor')
    {
        $data['doctor_id'] = $this->session->userdata('login_user_id');
    }
    else
    {
        $data['doctor_id'] = $this->input->post('doctor_id');
    }

    $this->db->insert('appointments',$data);

    // Notify patient with sms.
    $notify = $this->input->post('notify');
    if($notify != '') {
        $patient_name = $this->db->get_where('patient',
            array('patient_id' => $data['patient_id']))->row()->name;
        $doctor_name = $this->db->get_where('doctor',
            array('doctor_id' => $data['doctor_id']))->row()->name;
        $date = date("d, M Y", $data['timestamp']);
        $time = date("g:G a", $data['timestamp']);
        $message = $patient_name . ", you have an appointment with doctor: " . $doctor_name . " on " . $date . " at " . $time . " ";
        $receiver_phone = $this->db->get_where('patient',
            array('patient_id' => $data['patient_id']))->row()->phone;

        $this->sms_model->send_sms($message, $receiver_phone);
    }
}

```

Enviar SMS

O usuário Clickatell, a senha e o api_id são obtidos do banco de dados e uma chamada de autenticação é feita para verificar se o usuário é válido. Se for válido, um sms é enviado para o número do telefone do receptor com a mensagem que foi transmitida para a função como parâmetros.

```

function send_sms($message = '0', $receiver_phone = '0') {

    $clickatell_user = $this->db->get_where('settings', array('type' => 'clickatell_user'))->row()->description;
    $clickatell_password = $this->db->get_where('settings', array('type' => 'clickatell_password'))->row()->description;
    $clickatell_api_id = $this->db->get_where('settings', array('type' => 'clickatell_api_id'))->row()->description;
    $clickatell_baseurl = "http://api.clickatell.com";

    $text = urlencode($message);
    $to = $receiver_phone;

    // auth call
    $url = $clickatell_baseurl."/http/auth?user=$clickatell_user&password=$clickatell_password&api_id=$clickatell_api_id";

    // do auth call
    $ret = file($url);

    // explode our response. return string is on first line of the data returned
    $resp = explode("\n", $ret[0]);
    print_r($resp); echo "<br>";
    if ($resp[0] == "OK") {
        $sess_id = trim($resp[1]); // remove any whitespace.
        $url = $clickatell_baseurl."/http/sendmsg?session_id=$sess_id&to=$to&text=$text";

        // do sendmsg call
        $ret = file($url);
        $send = explode("\n", $ret[0]);
        print_r($send); echo "<br>";
        if ($send[0] == "ID") {
            echo "=====Message ID=====" . $send[1];
        } else {
            echo "Send message failed";
        }
    } else {
        echo "Authentication failure"; //... $ret[0];
    }
}

```

Mensagens Privadas

```

/* private messaging */

function message($param1 = 'message_home', $param2 = '', $param3 = '')
{
    if ($this->session->userdata('doctor_login') != 1)
        redirect(base_url(), 'refresh');

    if ($param1 == 'send_new') {
        $message_thread_code = $this->crud_model->send_new_private_message($param2);
        $this->session->set_flashdata('message', get_phrase('message_sent'));
        redirect(base_url() . 'index.php?doctor/message/message_read/' . $message_thread_code);
    }

    if ($param1 == 'send_reply') {
        $this->crud_model->send_reply_message($param2); // $param2 = message_thread_code
        $this->session->set_flashdata('message', get_phrase('message_sent'));
        redirect(base_url() . 'index.php?doctor/message/message_read/' . $message_thread_code);
    }
}

```

A mensagem de função contém as expressões lógicas para enviar novas mensagens, respondendo a qualquer thread de mensagem existente entre qualquer paciente que tenha consultas com o médico registrado.

As mensagens são rastreadas pelo código de linha da mensagem que está definido na função `send_new_private_message` localizada em `/application/models/crud_model.php`.

```
function send_new_private_message() {
    $message = $this->input->post('message');
    $timestamp = strtotime(date("Y-m-d H:i:s"));

    $receiver = $this->input->post('receiver');
    $sender = $this->session->userdata('login_type') . '-' . $this->session->userdata('login_user');

    //check if the thread between those 2 users exists, if not create new thread
    $num1 = $this->db->get_where('message_thread', array('sender' => $sender, 'receiver' => $receiver));
    $num2 = $this->db->get_where('message_thread', array('sender' => $receiver, 'receiver' => $sender));

    if ($num1 == 0 && $num2 == 0) {
        $message_thread_code = substr(md5(rand(100000000, 2000000000)), 0, 15);
        $data_message_thread['message_thread_code'] = $message_thread_code;
        $data_message_thread['sender'] = $sender;
        $data_message_thread['receiver'] = $receiver;
        $this->db->insert('message_thread', $data_message_thread);
    }
    if ($num1 > 0)
        $message_thread_code = $this->db->get_where('message_thread', array('sender' => $sender, 'receiver' => $receiver));
    if ($num2 > 0)
```

Accountant Controller (/application/controllers/accountant.php)

Adicionar fatura

A função `invoice_add` contém as expressões lógicas para criar uma nova fatura.

Isso é feito chamando a função `create_invoice` que está definida em `/ application / models / crud_model.php`. Um número de fatura aleatório é gerado para cada fatura. Várias entradas de fatura podem ser criadas para uma única fatura.


```

// Create a new invoice.
function create_invoice()
{
    $data['title'] = $this->input->post('title');
    $data['invoice_number'] = $this->input->post('invoice_number');
    $data['patient_id'] = $this->input->post('patient_id');
    $data['creation_timestamp'] = $this->input->post('creation_timestamp');
    $data['due_timestamp'] = $this->input->post('due_timestamp');
    $data['vat_percentage'] = $this->input->post('vat_percentage');
    $data['discount_amount'] = $this->input->post('discount_amount');
    $data['status'] = $this->input->post('status');

    $invoice_entries = array();
    $descriptions = $this->input->post('entry_description');
    $amounts = $this->input->post('entry_amount');
    $number_of_entries = sizeof($descriptions);

    for ($i = 0; $i < $number_of_entries; $i++)
    {
        if ($descriptions[$i] != "" && $amounts[$i] != "")
        {
            $new_entry = array('description' => $descriptions[$i], 'amount' => $amounts[$i]);
            array_push($invoice_entries, $new_entry);
        }
    }
    $data['invoice_entries'] = json_encode($invoice_entries);

    $this->db->insert('invoices', $data);
}

```

Controlador Recepcionista (/application/controllers/receptionist.php)

Gerenciar compromissos

```

function appointment($task = "", $doctor_id = 'all', $start_timestamp = "", $end_timestamp = "
[
    if ($this->session->userdata('receptionist_login') != 1)
    {
        $this->session->set_userdata('last_page' , current_url());
        redirect(base_url(), 'refresh');
    }

    if ($task == 'filter')
    {
        $doctor_id = $this->input->post('doctor_id');
        $start_timestamp = strtotime($this->input->post('start_timestamp'));
        $end_timestamp = strtotime($this->input->post('end_timestamp'));
        redirect('index.php?receptionist/appointment/search/' . $doctor_id . '/' . $start_time
    }

    if ($task == "create")
    {
        $this->crud_model->save_appointment_info();
        $this->session->set_flashdata('message' , get_phrase('appointment_info_saved_successfu
        redirect('index.php?receptionist/appointment');
    }

    $data['doctor_id'] = $doctor_id;
    if($start_timestamp == '')

```

A função de compromisso contém as expressões lógicas de mostrando todas as nomeações, criando um novo compromisso e informações de consulta de filtragem com base no médico e no tempo períodos.

Por nome próprio, os compromissos dos últimos 30 dias para todos os médicos são exibidos. Os parâmetros definem a ação do formulário em que os dados devem ser criados ou filtrados. Um sms é enviado para o paciente quando um novo compromisso é criado pelo chamado função send_sms com mensagem e número de telefone do destinatário como parâmetros definidos em / application / models / sms_model.php.

Aprovar compromissos

```
function approve_appointment_info($appointment_id)
{
    $data['timestamp'] = strtotime($this->input->post('date_timestamp')).' '.$this->input->post('time_timest
    $data['status'] = 'approved';

    if($this->session->userdata('login_type') == 'receptionist')
        $data['doctor_id'] = $this->input->post('doctor_id');

    $this->db->where('appointment_id', $appointment_id);
    $this->db->update('appointment', $data);

    // Notify patient with sms.
    $notify = $this->input->post('notify');
    if($notify != '') {
        $doctor_id = $this->db->get_where('appointment',
            array('appointment_id' => $appointment_id))->row()->doctor_id;
        $patient_id = $this->db->get_where('appointment',
            array('appointment_id' => $appointment_id))->row()->patient_id;
        $patient_name = $this->db->get_where('patient',
            array('patient_id' => $patient_id))->row()->name;
        $doctor_name = $this->db->get_where('doctor',
```

a função de compromisso_confirmada contém as expressões lógicas de mostrar todos os compromissos solicitados e aprovar um compromisso. Os parâmetros definem a forma de ação em que os dados devem ser aprovados. Um sms é enviado ao paciente quando um compromisso solicitado é aprovado chamando a função send_sms com o número de telefone da mensagem e do receptor como parâmetros definidos em / application / models / sms_model.php

Controlador de login (/application/controllers/login.php)

função ajax_login contém as expressões lógicas para

- correspondência de e-mail e senha para diferentes usuários
- configurando o tipo de login para diferentes usuários
- definir dados de sessão que são usados para outras visualizações

função reset_password

- corresponde ao endereço de e-mail para o endereço de e-mail salvo
- gera uma nova senha
- envia e-mail para esse endereço com a senha recém-gerada

log de função

- desactiva a sessão atual e destrói todos os dados da sessão salvos para essa sessão

Os Model Files (/ application / models)

- crud_model.php

contém as funções básicas para criar, recuperar, atualizar e excluir o que funciona em associação com controladores e visualizações.

- email_model.php

contém as funções que enviam e-mails em eventos diferentes.

- sms_model.php

contém a função que envia sms em eventos diferentes.

Os arquivos de exibição (/ application / views)

- backend

- admin

contém todos os arquivos de exibição do painel de administração - médico

contém todos os arquivos de exibição do painel de médicos - paciente

contém todos os arquivos de exibição do painel do paciente

- enfermeira

contém todos os arquivos de exibição do painel de enfermagem - farmacêutico

contém todos os arquivos de exibição do painel de farmacêutico - laboratorista

contém todos os arquivos de exibição do painel laboratorista - contador

contém todos os arquivos de exibição do painel contador - recepcionista

contém todos os arquivos de exibição do painel do recepcionista - footer.php

contém as informações do rodapé do aplicativo - header.php

contém a visão de cabeçalho do aplicativo - includes_bottom.php

contém links para arquivos javascript

- includes_top.php

contém links para arquivos css

- index.html

- index.php

inclui o rodapé, cabeçalho, javascript, css, modais no momento da página de carregamento

- login.php

a página de login

- modal.php

contém as vistas modal que são usadas na vista de diferentes painéis