dramis Documentation

Versión 0.0.1

Elvis Reyes

25 de agosto de 2018

Índice general

Nota: La presente documentación ha sido realizada enfocada en las nuevas actualizaciones requeridas por el cliente en el módulo referidos

CAPÍTULO 1

User Guide

1.1 Overview

1.1.1 Requerimientos

Ambiente de desarrollo.

- Vagrant v 2.0.1
- Virtualbox v 5.0.20
- PHP 5.6.32
- MariaDB `Server v: 10.0.33-MariaDB MariaDB Server`
- Git `v 1.7.1`
- Composer `v 1.6.2`
- Node `v 6.12.3` (Opcional)
- Npm `v 3.10.10` (Opcional)
- Gulp `v 2.0.0` (Opcional)

1.1.2 Instalación

Repositorio

Asumiendo que se tiene el ambiente de desarrollo configurado sólo se necitará lanzar el siguiente comando dentro del servidor de desarrollo en la carpeta principal de la aplicación.

```
cd /var/www/html/dramis
```

Si no se cuenta con el repositorio ven_develop se deberá crear una rama a partir de la rama remota del mismo con el siguiente comando:

```
git checkout -b ven_develop origin/ven_develop
```

Esto hará que se obtenga la rama `ven_develop` en el repositorio local.

Si, ya se cuenta con la rama `ven_develop` sólo se lanzará lanzará el siguiente comando:

```
git pull
```

El comando obtendrá los cambios realizados en la rama `ven_develop`

Nota: La versión del repositorio debe ser `ven_develop`

Node (Opcional)

Para instalar node en el sistema siga los pasos de su página oficial Node `v 6.12.3`

Npm (Opcional)

Para instalar npm de manera global siga los pasos en su página oficial ó ejecute el siguiente comando para realizar una instalación global

```
npm install npm@latest -g
```

1.1.3 Actualización base de Datos

authorizations

```
ALTER TABLE `authorizations`
ADD COLUMN `followupdate` DATETIME NULL DEFAULT NULL AFTER `referral_id`,
ADD COLUMN `statusDate` DATETIME NULL DEFAULT NULL AFTER `followupdate`,
ADD COLUMN `reqDate` DATETIME NULL DEFAULT NULL AFTER `statusDate`,
ADD COLUMN `supervisor` VARCHAR(245) NULL DEFAULT NULL AFTER `reqDate`,
ADD COLUMN `provider` VARCHAR(245) NULL DEFAULT NULL AFTER `supervisor`,
ADD COLUMN `created_by` VARCHAR(245) NULL DEFAULT NULL AFTER `provider`,
ADD COLUMN `worked_date` DATETIME NULL DEFAULT NULL AFTER `created_by`;
```

fields

form_type `(read)`

authorization `(read/write)`

total_units_left `(read)`

end_date `(read)`

patient_id `(read)`

insurance_id `(read)`

status `(read)`

updated_at (read/write)

updated_by (read/write)

auth_comments (read/write)

referral_id (read)

followupdate (read/write)

statusDate (read/write)

reqDate (read/write)

created_by (read/write)

worked_date (read/write)

referrals (new)

```
CREATE TABLE `referrals` (
`id` BIGINT(20) NOT NULL AUTO_INCREMENT,
`first_name` VARCHAR(255) NULL DEFAULT NULL,
`last_name` VARCHAR(255) NULL DEFAULT NULL,
`full_name` VARCHAR(255) NULL DEFAULT NULL,
`parental_guardian_name` VARCHAR(255) NULL DEFAULT NULL,
`date` DATETIME NULL DEFAULT NULL COMMENT 'referral_date',
`dob_date` VARCHAR(255) NULL DEFAULT NULL,
`sex` VARCHAR(255) NULL DEFAULT NULL,
`ethnic` VARCHAR(255) NULL DEFAULT NULL,
`race` VARCHAR(255) NULL DEFAULT NULL,
`ssn` VARCHAR(255) NULL DEFAULT NULL,
`home_phone` VARCHAR(255) NULL DEFAULT NULL,
`cell_phone` VARCHAR(255) NULL DEFAULT NULL,
`preferred_phone` VARCHAR(255) NULL DEFAULT NULL,
`county` VARCHAR(255) NULL DEFAULT NULL,
`address` VARCHAR(255) NULL DEFAULT NULL,
`city` VARCHAR(255) NULL DEFAULT NULL,
`state` VARCHAR(255) NULL DEFAULT NULL,
```

fields

all fields (read/write)

id bigint(20) AI PK

first_name varchar(255)

last_name varchar(255)

full_name varchar(255)

parental_guardian_name varchar(255)

date datetime

dob_date varchar(255)

sex varchar(255)

ethnic varchar(255)

race varchar(255)

ssn varchar(255)

home_phone varchar(255)

cell_phone varchar(255)

preferred_phone varchar(255)

county varchar(255)

address varchar(255)

city varchar(255)

state varchar(255)

zip_code varchar(255)

language_preference varchar(255)

school varchar(255)

grade varchar(255)

ese varchar(255)

client_id bigint(20)

service_required varchar(255)

reason_for_referral varchar(255)

other varchar(255)

current_treatment text

previous_treatment text

diagnosis text

medications text

physician_name varchar(255)

physician_phone varchar(255)

physician_fax varchar(255)

comments text

insurance_info varchar(255)

insurance_id varchar(255)

insurance_other_id varchar(255)

referral_full_name varchar(255)

referral_agency varchar(255)

email varchar(255)

referral_phone varchar(255)

referral_fax varchar(255)

referral_requested_therapist varchar(255)

referral_taken_by varchar(255)

reviewed `tinyint(2)`

worked `tinyint(2)`

eligible `tinyint(2)`

created_at `timestamp`

updated_at `timestamp`

needy_service `varchar(255)`

closed `tinyint(2)`

referral_comments `text`

client_type `varchar(255)`

creation_assigned_to `varchar(255)`

creation_date `datetime`

patient_id `int(50)`

evaluationReq `tinyint(2)`

TherapyCenter `varchar(255)`

TherapyCenterSAssignedDate `datetime`

verified_by `varchar(255)`

created_by_user `tinyint(2)`

referred_by `varchar(255)`

referral_services (new) all fields read/write

```
CREATE TABLE `referral_services` (
`id` BIGINT(20) NOT NULL AUTO_INCREMENT,
`name` VARCHAR(145) NULL DEFAULT NULL,
`prefix` VARCHAR(45) NULL DEFAULT NULL,
PRIMARY KEY (`id`),
UNIQUE INDEX `name_UNIQUE` (`name` ASC),
UNIQUE INDEX `prefix_UNIQUE` (`prefix` ASC));
```

fields

id `bigint(20) AI PK`

name `varchar(145)`

prefix `varchar(45)`

patient_staff_assignment all fields read

fields

patient_staff_id `int(12) AI PK`

patient_id `int(12)`

staff_id `int(12)`

staff_title `varchar(250)`

status `int(1)`

treatment `int(1)`

trans_id `int(11)`

assigned_by `varchar(45)`

date `timestamp`

insurance_companies all fields read

id `int(11) AI PK`

name `varchar(255)`

attn `varchar(255)`

cms_id `varchar(15)`

freeb_type `tinyint(2)`

x12_receiver_id `varchar(25)`

x12_default_partner_id `int(11)`

alt_cms_id `varchar(15)`

patient_data all fields read

-

Modificación de base de datos (alters)

Se ha resuelto agregar un campo en la base de datos llamado «disabled» en la tabla «authorizations». Con este nuevo ajuste se podrá lograr el funcionamiento esperado. A continuación se presenta el query de este cambio

```
ALTER TABLE `authorizations`
ADD COLUMN `disabled` TINYINT(1) NULL DEFAULT 0 AFTER `worked_date`;
```

1.1.4 Inventario de archivos modificados

En esta versión se proporcina la lista de los siguientes archivos modificados:

html/
├─ api/
│ ├─ index.php
│ ├─ controllers/
│ ├── Authorization.php
│ ├── ClinicReferral.php
│ ├── WpFormController.php

```
│ ├─ models/
│ ├── Authorization.php
│ ├── Patient.php
│ ├── Staff.php
│ ├── ReferralService.php (Nuevo)
│ └── Referral.php (Nuevo)
│
├─ interface/
│ ├─ patient_file/
│ │ ├─ summary/
│ │ ├── authorizations_full.php
│ │ └── record_authorization.php
│ │
│ ├─ main/
│ │ └── left_nav.php
│ ├─ wp_forms/
└ └──── *
```

Nota: Las descripciones de los códigos actualizados se podrán observar en el historial de git.

1.1.5 FrontEnd Compiler

El frontend se ha recompilado con el gestor de tareas gulp, este proceso totalmente opcional y se realiza en tal caso cuando al hacer un `git pull` el servidor no reconozca los cambios del frontend.

Los motivos por los que se ha utilizado dicho gestor es debido a que cuando los archivos `.js` en la carpeta `/var/www/html/dramis/html/interface/wp_forms/js` eran modificados, el servidor no detectaba los cambios.

Se decidió usar gulp para generar una nueva versión del archivo y obligar al servidor que liste el nuevo archivo.

El proceso consiste en que el gestor de tareas gulp genera un `rev-manifest.json` en la carpeta `/var/www/html/dramis/html/interface/wp_forms/assets`

```
{
"app.js": "app-0806c0cadf.js",
"camdComponent.js": "camdComponent-80bd8a0091.js",
"cbhssamComponent.js": "cbhssamComponent-f99c929ecf.js",
"csamComponent.js": "csamComponent-c9e1291a62.js",
"referraldetails.js": "referraldetails-493f1167ff.js"
}
```

En este ejemplo se puede observar un json de tipo `key-value` donde:

- key: representa el nombre del archivo modificado. `"app.js"`
- value: representa la versión del archivo. `"app-0806c0cadf.js"`

En la carpera `wp_forms` vienen incluidos los archivos de configuración del gestor de tareas `gulp`. Estos archivos son:

- gulpfile.js (archivo de configuración)

- package.json (gestor de paquetes)

En caso de que se requiera compilar los archivos `.js` localizados en wp_forms/js/* en primera instacia se deberá activar la instalación de gulp mediante npm ejecutando el siguiente comando detro de la carpeta `wp_forms`

```
npm install --save
```

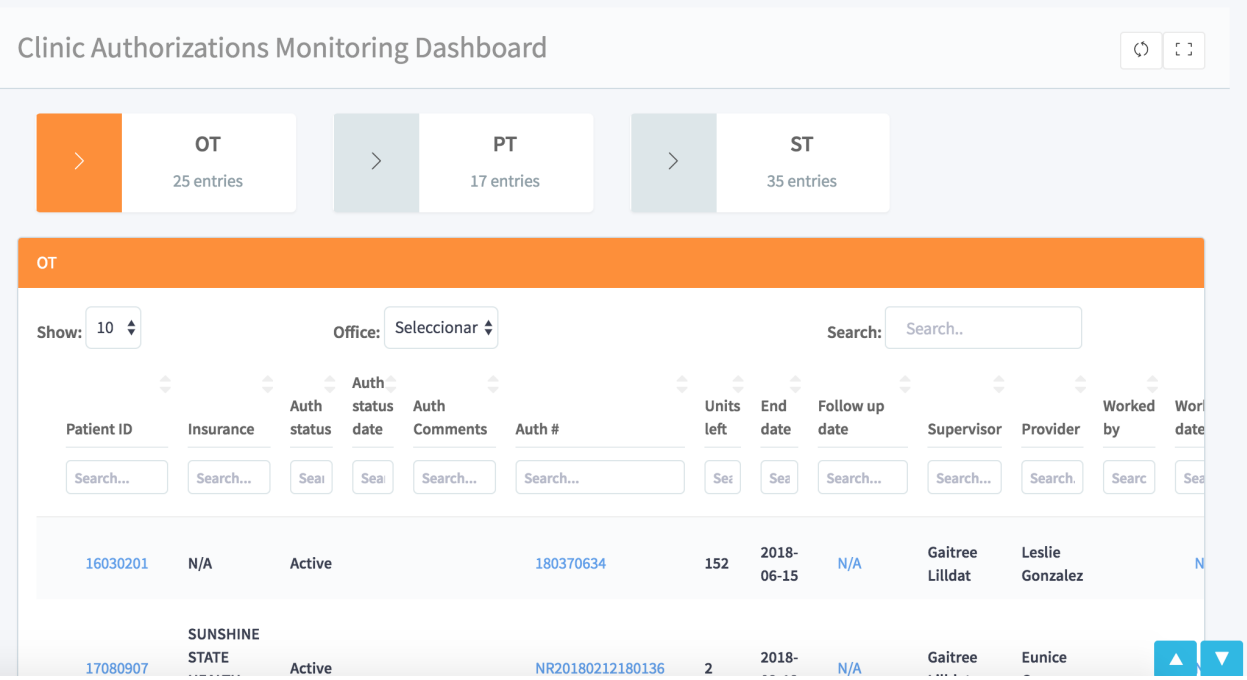
Esto hará que se descarguen todas las depenencias necesarias a nivel local para que el gestor de tareas gulp funcione.

Después de activar la instalación sólo se deberá ejecutar el siguiente comando para compilar los archivos js modificados

```
npm run build
```

1.2 Modules

1.2.1 Clinic Authorizations Monitoring Dashboard



Requisitos iniciales

Las autorizaciones que se deben ver en este dashboard son las siguientes:

- Autorizaciones activas que les quede menos de 8 unidades
- Autorizaciones activas que vencen en los próximos 15 días
- El “dashboard” debe estar dividido por especialidad OT, PT, ST
- La autorización en el “dashboard” debe tener un link que lleve directo a las autorizaciones del paciente
- Debe tener una columna donde se pueda poner la fecha en que se trabajó la autorización y autómaticamente deje de aparecer en el dashboard
- Las columnas que debe tener son las siguientes

- Debemos poder ver todas las transacciones por localidad de ser necesario (Davenport, St.Cloud)

Se presentan los archivos incolucrados en este dashboard.

FrontEnd

- `/html/interface/wp_forms/index.php`
- `/html/interface/wp_forms/js/app.js`
- `/html/interface/wp_forms/js/view-components/camdComponent.js`
- `/html/interface/patient_file/summary/authorizations_full.php`
- `/html/interface/patient_file/summary/record_authorization.php`

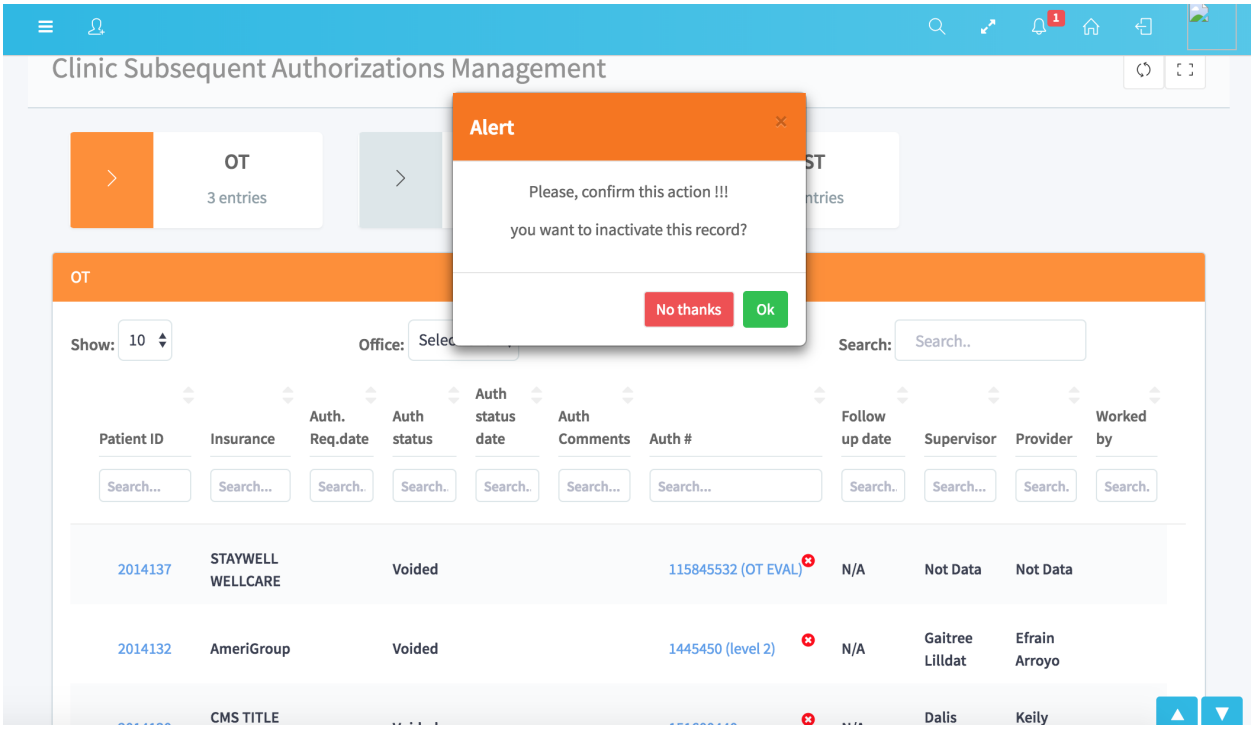
Backend

- `/html/api/index.php`
- `/html/api/controllers/WpFormController.php`
- `/html/api/modeles/Authorization.php`

Nota: Si se desea compilar los archivos del FrontEnd deberá ejecutarse el siguiente comando en la carpeta `/html/interface/wp_forms/`

```
npm run build
```

1.2.2 Clinic Subsequent Authorizations Management



Requisitos iniciales

- Este dashboard debe presentar todas las autorizaciones que no estén atadas a un referido y que tengan cualquier estatus menos activa e inactiva.
- La autorización en el “dashboard” debe tener un link que lleve directo a las autorizaciones del paciente
- El “dashboard” debe estar dividido por especialidad OT, PT, ST
- Las columnas que debe tener son las siguientes
- Debemos poder ver todas las transacciones por localidad de ser necesario (Davenport, St.Cloud)

Se presentan los archivos incolucrados en este dashboard.

FrontEnd

- `/html/interface/wp_forms/index.php`
- `/html/interface/wp_forms/js/app.js`
- `/html/interface/wp_forms/js/view-components/csamdComponent.js`
- `/html/interface/patient_file/summary/authorizations_full.php`
- `/html/interface/patient_file/summary/record_authorization.php`

Backend

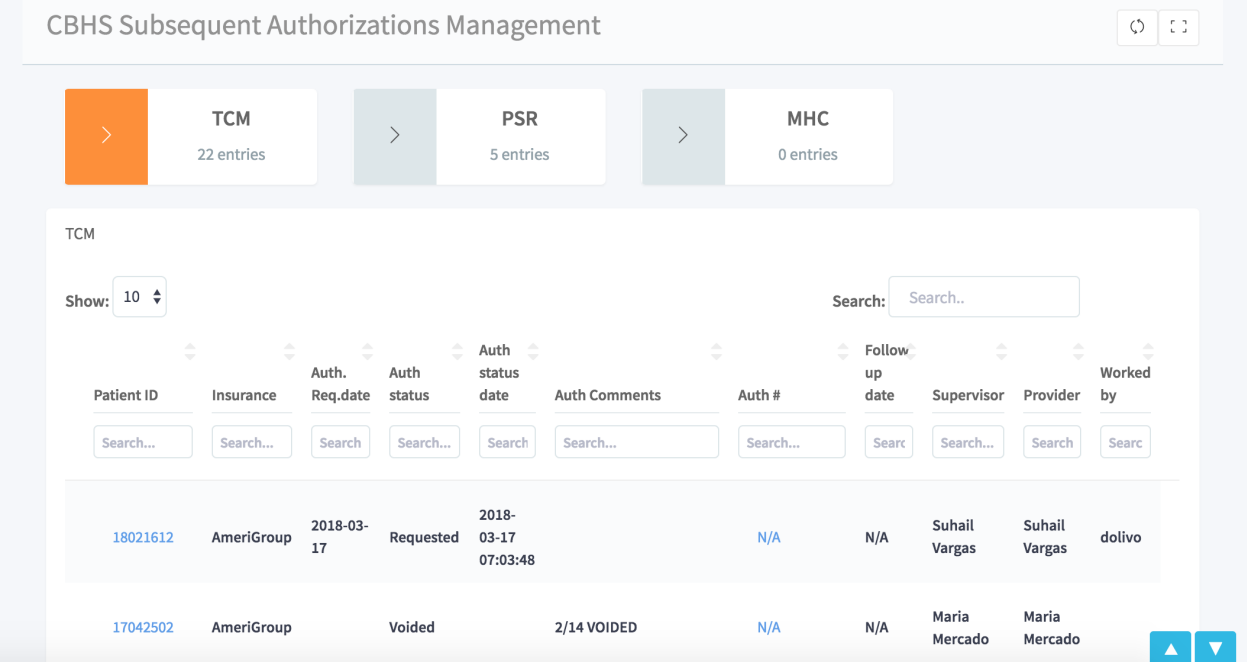
- `/html/api/index.php`

- `/html/api/controllers/WpFormController.php`
- `/html/api/modeles/Authorization.php`

Nota: Si se desea compilar los archivos del FrontEnd deberá ejecutarse el siguiente comando en la carpeta `/html/interface/wp_forms/`

```
npm run build
```

1.2.3 CBHS Subsequent Authorizations Management



Requisitos iniciales

- Este dashboard debe presentar todas las autorizaciones que no estén atadas a un referido y que tengan cualquier estatus menos activa e inactiva
- La autorización en el "dashboard" debe tener un link que lleve directo a las autorizaciones del paciente
- El "dashboard" debe estar dividido por especialidad TCM, PSR, MHC
- Las columnas que debe tener son las siguientes
- Debemos poder ver todas las transacciones por localidad de ser necesario (Davenport, St.Cloud) (Es una columna adicional o un filtro)
- Poner como editable todas las columnas que se puedan editar y que los cambios puedan actualizar el récord del paciente.

Se presentan los archivos incolucrados en este dashboard.

FrontEnd

- `/html/interface/wp_forms/index.php`
- `/html/interface/wp_forms/js/app.js`
- `/html/interface/wp_forms/js/view-components/cbhssamComponent.js`
- `/html/interface/patient_file/summary/authorizations_full.php`
- `/html/interface/patient_file/summary/record_authorization.php`

Backend

- `/html/api/index.php`
- `/html/api/controllers/WpFormController.php`
- `/html/api/modeles/Authorization.php`

Nota: Si se desea compilar los archivos del FrontEnd deberá ejecutarse el siguiente comando en la carpeta `/html/interface/wp_forms/`

```
npm run build
```

1.2.4 CBHS Subsequent Authorizations Management (Corrigiendo funcionalidades)

Requisitos iniciales

- Este dashboard debe presentar todas las autorizaciones que no estén atadas a un referido y que tengan cualquier estatus menos activa e inactiva
- La autorización en el "dashboard" debe tener un link que lleve directo a las autorizaciones del paciente
- El "dashboard" debe estar dividido por especialidad TCM, PSR, MHC
- Las columnas que debe tener son las siguientes
- Debemos poder ver todas las transacciones por localidad de ser necesario (Davenport, St.Cloud) (Es una columna adicional o un filtro)
- Poner como editable todas las columnas que se puedan editar y que los cambios puedan actualizar el récord del paciente.

Correciones

- Columna de "auth requested date" no va en este dashboard
- Faltaron las siguientes columnas "units left, "end date" y "worked date"

Se presentan los archivos incolucrados en estla correción de este dashboard.

FrontEnd

- `html/interface/wp_forms/assets/rev-manifest.json`
- `html/interface/wp_forms/index.php`
- `/html/interface/wp_forms/js/view-components/cbhssamComponent.js`

Nota: Si se desea compilar los archivos del FrontEnd deberá ejecutarse el siguiente comando en la carpeta `/html/interface/wp_forms/`

```
npm run build
```
