pharmacy duty roster
Documentation

Martin Mandelkow

April 24, 2019

Contents

Chapter 1

Introduction

Pharmacy Duty Roster (PDR) is a web application that allows to operate a duty roster for pharmacies. PDR started in 2015 as an alternative to a really simple excel sheet without formulas. PDR aims to be user-friendly but at the same time cover all necessary features. PDR continuously strives to improve. It is open to your requests and wishes. I hope it will fulfil your expectations.

1.1 Getting PDR

The latest release of PDR is available on GitHub. GitHub provides the source code as *.zip file or *.tar.gz ball. Extract the files into a folder.

Make sure to unpack PDR to a directory, that your webserver has access to. PHP and the webserver must have read access to all the files and folders. It also needs write access to the subdirectories upload, tmp and config. You might want to change the owner of the directory to the webservers user with e.g.:

```
sudo chown -R www-data:www-data /var/www/html/pdr/
```

You can also clone the repository with git:

```
git clone https://github.com/MaMaKow/dienstplan-apotheke.git
```

See the Administrator manual for details!

1.2 License

PDR is open source software under the AGPL license.

> You should have received a copy of the GNU Affero General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Please see the license file for details!

1.3 Reporting bugs

The issue tracker is currently located at GitHub `https://github.com/MaMaKow/dienstplan-apotheke/issues`. GitHub requires an account in order to report bugs or feature requests. If you do not want to create one, you might send a mail to pdr-issues@martin-mandelkow.de

1.4 How to contribute

Pull requests are desired. If you made changes to PDR and want to contribute them to the public, you are welcome to open a pull-request on GitHub or send your changes in any other way.

You might as well use `git send-email` and send patches to pdr-discuss@martin-mandelkow.de

Chapter 2

User manual

2.1 The web interface

You can connect to your PDR instance using any web browser. Just navigate to your server and enter your username and password.

2.1.1 Login

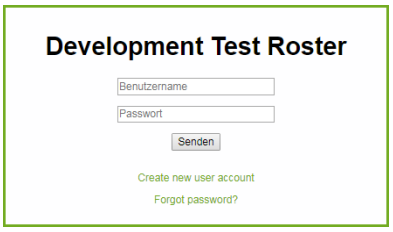


Figure 2.1: Login page

The login page shows the name of the application. You are prompted to enter your username and password. If you do not have an account yet, you can Create a new user account. If you have an account, but forgot about your password, or want to change it, you can click on Forgot password?.

2.1.2 Create new user account

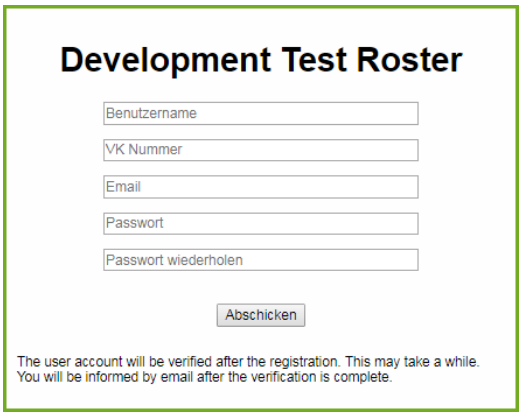


Figure 2.2: Register new user page

Choose a user name, enter your employee id and your email. Pick a secure password.

The account will be inactive until an administrator activates it. The main administrator is informed via email regarding the registration.

New users can only be created for existing employees. New employees are created by an administrator.

2.1.3 Lost password

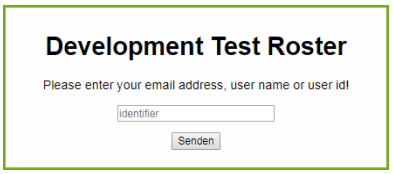


Figure 2.3: Lost password page

The lost password page shows the name of the application. You are prompted to enter either your username, id or your email-address at your option. After you submit the form, an email is sent to your stored email address. In that email you will find a link, which will lead you to the password change page.

Lost password recovery

Figure 2.4: Lost password recovery page

The lost password recovery page shows the name of the application and your user name. You are prompted to enter a new password twice.

2.1.4 Navigation

Figure 2.5: Navigation bar

By default, the PDR web interface opens a menu containing 5 tiles. You can navigate to:

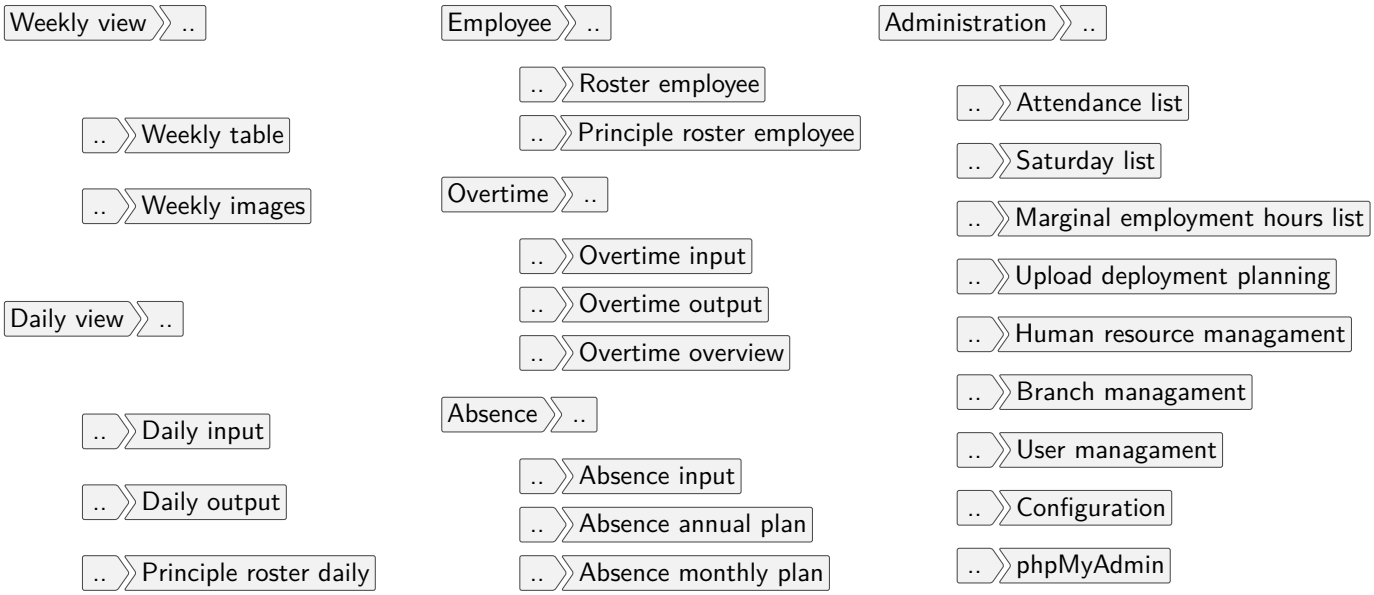
- Roster week table view 
- Roster daily view 

- Roster employee view 

- Overtime 
- Absence 

The navigation bar

In the top there is a navigation bar containing hyperlinks to nearly all the pages of PDR. Hover the mouse over an entry to open the submenus (Figure 2.5).



2.1.5 Roster week table view

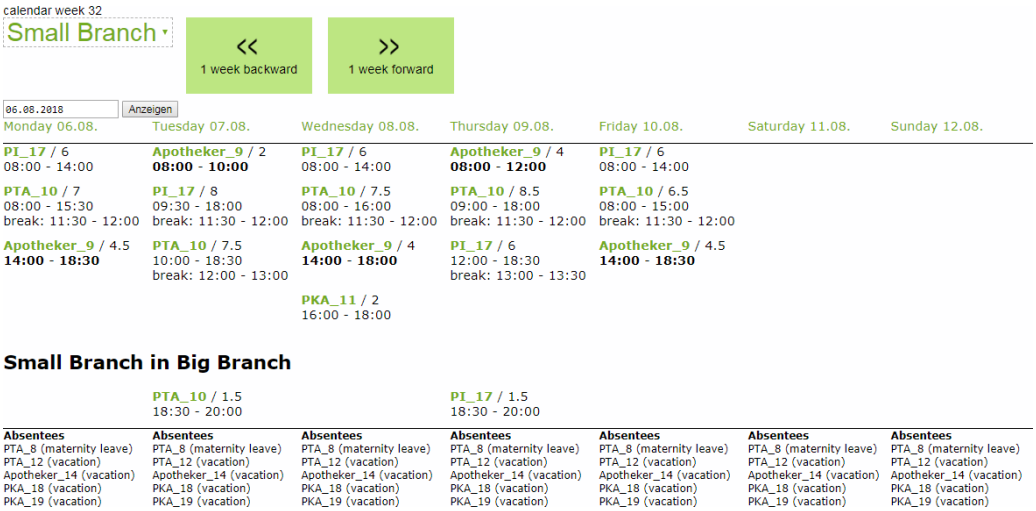


Figure 2.6: Roster week table view, excerpt without task rotation and weekly working hours

The roster week table view shows the roster of a chosen week and branch (Figure 2.6). If employees of the branch are working in an other branch, then those are shown below. The table foot contains the information about absent employees and their reason of absence.

The date can be chosen by direct input. It can also be shifted by one week backwards or forwards by pressing Ctrl + ⇧ + → or Ctrl + ⇧ + ← respectively.

2.1.6 Roster daily view

Read only

In the daily roster view there is a table, a bar plot and a histogram reflecting the roster (Figure 2.7).

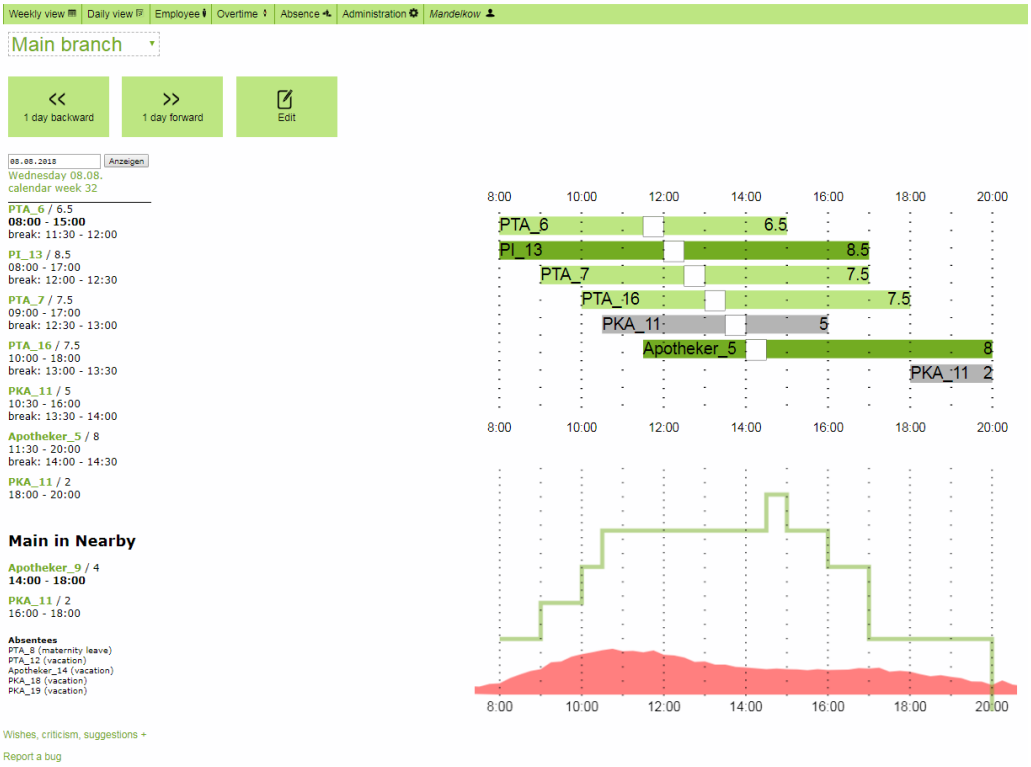


Figure 2.7: Roster day view

The roster table lists all the employees scheduled in the chosen branch on the one chosen day. For every entry there is the employee id and last name, the working hours, the start and end of duty and the time of the lunch break, if any.

If an employee, that is primarily scheduled in the chosen branch, works in an other branch, then this entry is shown in the table at the bottom. An employee may have more than one entrie per day. This allows divided working time to be stored. If employees are absent, these absences are displayed in the table footer.

The roster bar plot shows the flow of employees coming and going. Each bar represents one entry. It reaches from the start of duty to its end. A white rectangle on the bar shows the time of the lunch break. The color of the bars is dependent on the profession of the employee. Pharmacists and Pharmazieingenieure [1] are colored in dark green, while Pharmacy technicians are colored in light green. Other employees (non-pharmaceutical personnel) are colored in grey.

The histogram plot shows a red area and a green line. The red area shows the expected amount of work (measured in packages per 15 minutes), while the green line represents the amount of working employees on any given time.

Edit

The edit page looks quite similar to the read only view (Figure 2.8).

The roster is examined for errors. If any issues occur, then errors, warnings or information will be shown in the top right area. The examination includes:

[1] specific eastern german profession, see https://de.wikipedia.org/wiki/Pharmazieingenieur

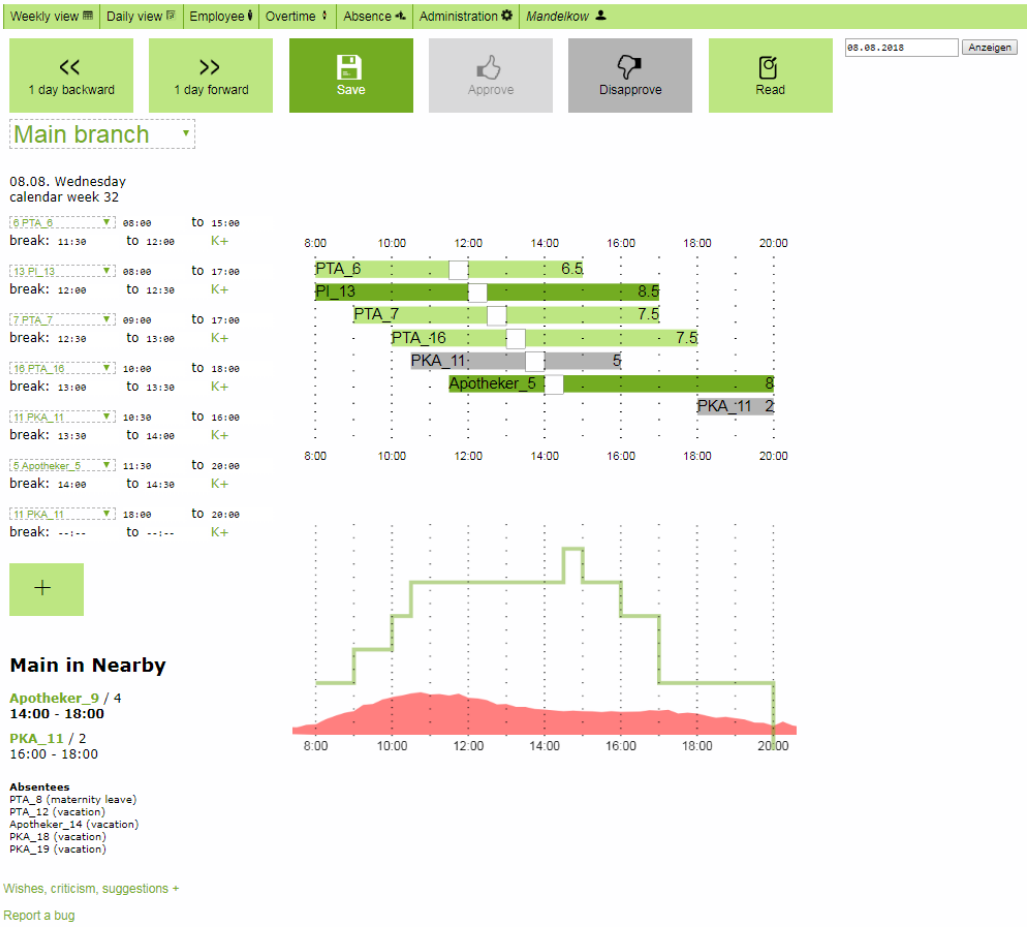


Figure 2.8: Roster day view with edit privileges

- overlap of shifts for the same employee (Error)
- sufficient employee count (Warning, hardcoded at least two employees)
- attendance of at least one pharmacist at any time (Error).
- attendance of at least one person able to carry out goods receipt (Warning).
- scheduling of absent employees (Error)
- non-scheduling of non-absent employees (Warning)

Only one break can be inserted per entry. If more breaks have to be assigned, then it is possible to enter multiple entries for the same employee.

2.1.7 Roster employee view

The employee view is similar to the weekly view, but only one employee is shown (Figure 2.9).

An iCalendar file can be downloaded. See section 2.2 Calendar API for details.

2.1.8 Principle roster daily

A principle recurring roster can be saved. In the simplest case it is a list of the start and end of duty for all the employees. Every weekday is listed separately, and so are the branches.

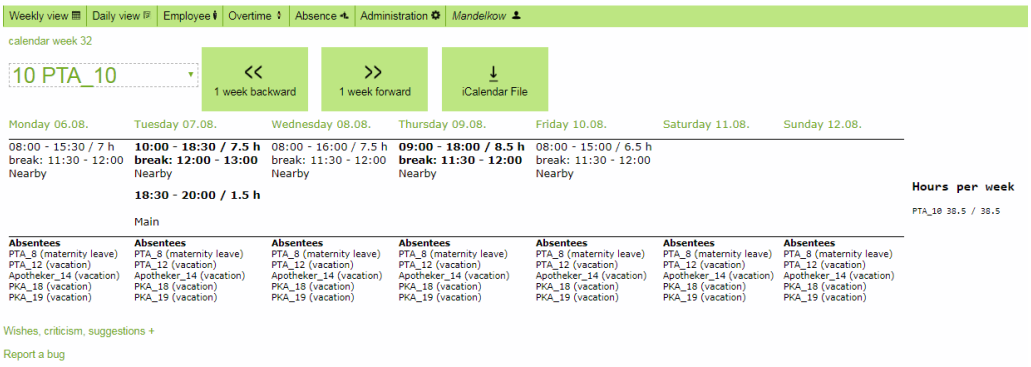


Figure 2.9: Employee view

It is possible to create alternating weeks. For example an employee might work on an A-Week and a B-Week. On A-Weeks she might regularly start at 08:00 in the morning on Mondays, while on B-Weeks she starts at 10:00.

2.1.9 Overtime

2.1.10 Absence

There are four views to the absence data.

- Employee view readonly
- Employee view edit
- Monthly table
- Year overview

In the *Employee view readonly* there is a select element to choose the employee to view. There is a button to switch to the edit view. And there is a table containing the absence data. The columns are start and end of the absence, reason of absence and number of days. There is a distinct list of possible reasons (vacation, remaining holiday, sickness, sickness of child, unpaid leave of absence, paid leave of absence, parental leave and maternity leave). The number of days of absence is calculated for a 5 day week. Absences on saturdays and sundays are registered but not counted. The same applys for holidays.

2.2 Calendar API

It is possible to read the roster data from PDR in form of iCalendar files. These files can be used with all major calendar applications on desktops and smartphones. This API is by no means a full implementation of the webdav standard. It is not even an implementation of the CalDAV protocol. Just point your browser to the following URL: `https://YOURHOSTNAME/YOUR/FOLDER/webdav.php`

The options are:

employee_id The employee id of the user from whom the roster should be given. Any user can get the roster of every employee. (default = the logged in user)

date_string A date in the format YYYY-MM-DD (default = today)

days_into_the_future The number of days that should be in the calendar file. (default = 30)

create_valarm Create an alarm (ACTION:DISPLAY) on your device. (default = 0)

- 0 = no alarm
- 1 = alarm 30 minutes before beginning of duty
- 2 = alarm on the end of duty
- 4 = alarm when the lunch break starts
- 8 = alarm when the lunch break ends
- 11 = 1+2+8 = alarm for start and end of duty and for end of lunch, but not for start of lunch

In order to get the roster of the week starting on 17.12.2018 for the employee number 5 you would use the following url: `https://YOURHOSTNAME/YOUR/FOLDER/webdav.php?employee_id=5&date_string=2018-12-17&days_into_the_future=6`

2.2.1 Automatically import iCalendar files with "iCal Import/Export CalDAV Pro"

Automatic import of iCalendar files into Android smartphones has been tested with "iCal Import/Export CalDAV Pro" (3,59 EUR). There are probably other apps, that can do the same.

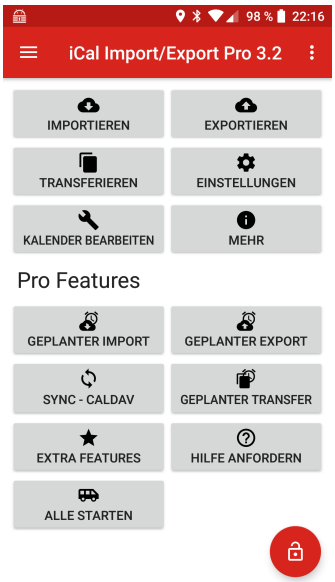


Figure 2.10: iCal main menu

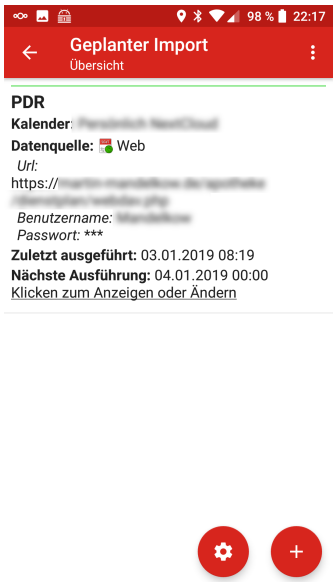


Figure 2.11: iCal planned imports

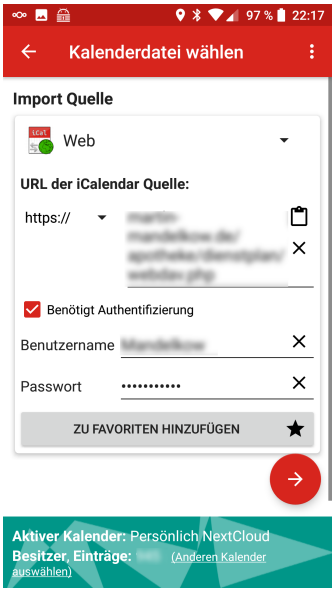


Figure 2.12: iCal adding a new import source

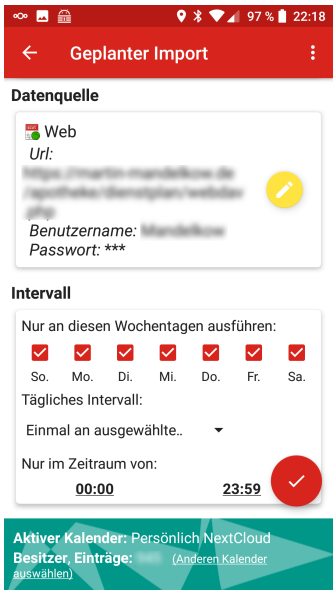


Figure 2.13: iCal setup import intervals

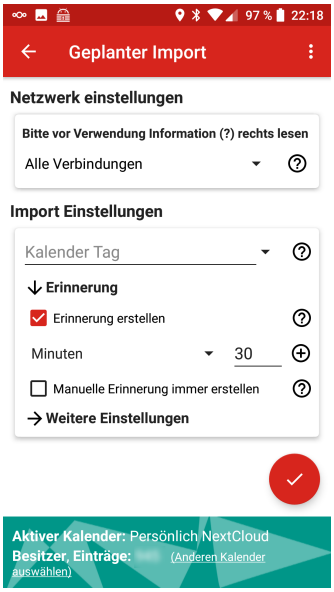


Figure 2.14: iCal network settings and import settings

Chapter 3

Administrator manual

3.1 Installation

3.1.1 Getting PDR

The latest release of PDR is available on GitHub

You can also get the latest stable version via git:

```
1 git clone https://github.com/MaMaKow/dienstplan-apotheke.git
```

The master branch is tested to be stable.

3.1.2 The installer

Introduction

The first page shows some non-technical information about this program. Click Next to move on.

Welcome

On the second page some technical background information is given. You are informed about the necessary input data, required for continuing the installation. Available database management systems (currently only MySQL) are listed. Finally, you are informed about the user and password strategy for the database access. Click Next again, to continue.

Requirements

On the next page the application checks, if all requirements are met. These include a minimum PHP version, some PHP extensions and support for database connections. Also the program needs write access to some of its directories. If problems are found, then a descriptive error message will be shown. It is not possible to continue, until all issues are solved. Click Next again, to continue.

Database configuration

The application now starts to collect configuration data.

- Database type

- hostname
- port (optional)
- username
  An existing database user. The user MUST have the privilege to create a database. The user SHOULD have the privilege to create a less privileged user.
- password
  The database password of the user. If a new user could be created, then a new secure random password will be given to the new user.
- database name

Enter the required data and Submit it.

Administrator configuration

After the database values are set, some information about the administrator is collected:

- User name
  the name used by the administrator to login into the program in the future.
- Last name
  this name is connected to the employee id.
- Employee id
  this is used to create an employee, who is connected to the administrative user.
- Contact email address
  is used for questions and comments from the users. Also this email will receive some internal information from the roster.
- Administrator password
  the password used by the administrator to login into the program in the future.

Please register the administrator and click Submit. The data will be written to the file `config/config.php`. For every user, that uses the program, there has be be exactly one employee.

3.1.3 First steps

After submitting the administrator configuration, you will be forwarded to the login page. Login with your administrator credentials.

On your first login you will be prompted with the branch management. Please create at least one branch. You can reach this page at all times in the menu Administration › Branch Management.

The next logical step is to setup some more employees in the Administration › Human resource management.

After all the employees are inserted, you can just start to write rosters (Daily View › Daily input) or you might create principle rosters for specific weekdays (Daily view › principle roster daily) or for distinct employees (Employee › Principle roster employee).

3.2 Upgrading

Until now, there is no automatic update mechanism established. You can regularly download release packages from GitHub. Or you can stay in touch via git:

```
git pull origin master
```

CAVE: Make sure, that you keep your `config/config.php`! It should not be changed by git, because it is listed in the .gitignore file of this project.

3.3 Configuration

You can manually edit the file `config/config.php`. The default values are:

```
<?php

$config = array(
        'application_name' => 'PDR',
        'database_management_system' => 'mysql',
        'database_host' => 'localhost',
        'database_name' => '',
        'database_port' => 3306,
        'database_user' => '',
        'database_password' => '',
        'session_secret' => '',
        'error_reporting' => E_ALL,
        'display_errors' => 0,
        'log_errors' => 1,
        'error_log' => PDR_FILE_SYSTEM_APPLICATION_PATH . 'error.log',
        'LC_TIME' => 'C',
        'timezone' => 'Europe/Berlin',
        'language' => 'de_DE',
        'mb_internal_encoding' => 'UTF-8',
        'contact_email' => '',
        'hide_disapproved' => FALSE,
        'email_method' => 'mail',
        'email_smtp_host' => NULL,
        'email_smtp_port' => 587,
        'email_smtp_username' => NULL,
        'email_smtp_password' => NULL,
);
```

Never delete the first two lines! If the file does not start with `<?php` then PHP will not handle it, meaning that anyone can read its content.

Most of these options can also be configured in Administration » Configuration

application name This name is used in the login page, the page title in the browser and as a subject line in emails sent from the program.

database settings

- database_management_system Currently only mysql is supported. Other possibilities could be: PostgreSQL, Oracle Database, SQLite, Microsoft Access or MongoDB.

- database_host The server running the DBMS. Usually 'localhost', if it is on the same server as the application.
- database_name The name of the database
- database_port For MySQL the standard port is 3306.
- database_user During the installation PDR will try to create the user 'pdr' in the database. In the case of success it will choose a random password and grant all privileges on the pdr database. You can choose any other user with access to the database.
- database_password The database password of the database user.

session_secret A secret random string used to define the session name. This is relevant only if multiple instances of PDR are running on the same webserver.

error_reporting Which errors should PHP report?

display_errors Should errors be directly displayed to the user?

log_errors Should errors be logged in a file?

error_log Where should errors be logged?

LC_TIME In wich language should time strings, such as Monday or January be written? This setting is independant from the 'language' setting.

timezone The timezone is necessary to make sense of the raw time data in unix time stamps. (1545038523 = Monday, 17-Dec-18 09:22:03 UTC in RFC 2822)

language The language of the terms and messages displayed to the user. Only English and German are supported until now.

mb_internal_encoding An encoding is a way to tell the computer how bits and bytes are translated into letters (e.g. in UTF8 01000101 means E, 11000011 10100100 means ä).

contact_email Emails of the users can be sent to the administrator of the pdr instance.

hide_disapproved It is possible, to hide scheduled rosters, until they are approved. By default all rosters are immediately visible to any user.

email settings Emails are sent (partly) with the PHPMailer class.

- 'email_method' PHPMailer supports sending via 'mail', 'sendmail', 'qmail' or 'SMTP'. If SMTP is chosen as the email method, then the SMTP settings will be displayed and have to be filled out:
- 'email_smtp_host' The address of the SMTP server (e.g. postfix on localhost, gmail-smtp-in.l.google.com for gmail)

- 'email_smtp_port' Normally one of 25, 465 or 587 (25 = SMTP, 465 = SMTPS, 587 = STARTTLS)
- 'email_smtp_username' username of the sending mail account
- 'email_smtp_password' password of the sending mail account

3.4 Maintenance

The file `src/php/background_maintenance.php` will be called on every login of any user. It will create an instance of the following classes:

- maintenance
- update_database

3.4.1 class maintenance

The methods contained in the class maintenance are only called, if the last execution is at least MAINTENANCE_PERIOD_IN_SECONDS ago. MAINTENANCE_PERIOD_IN_SECONDS is set to once a day.

The method user_dialog_email->aggregate_messages_about_changed_roster_to_employees() is called to send out emails to employees, whose roster has been changed.

The method maintenance->cleanup_overtime() does not do anything yet. It is meant to be used to clean up overtime of existing employees, that happened before they entered the company.

The method maintenance->cleanup_absence() does not do anything yet. It is meant to be used to clean up absences of existing employees, that happened before they entered the company.

3.4.2 class update_database

Before the class update_database executes any of its methods, it checks if there is any change in the official database structure since the last update. It compares the pdr_database_version_hash from the table pdr_self in the database with the value PDR_DATABASE_VERSION_HASH stored in the file `src/php/database_version_hash.php`

The class should contain every SQL Query necessary to change an existing database into the new structure without loosing any data.

3.5 Issues and Troubleshooting

Issues are tracked at GitHub https://github.com/MaMaKow/dienstplan-apotheke/issues
I am trying to answer any question within 3 days.

Chapter 4

Developer manual

4.1 Core development

All PHP scripts have a common file `default.php`, which handles the default settings. It is placed at ./, which is the `PDR_FILE_SYSTEM_APPLICATION_PATH`. See the file below:

```php
<?php

/*
 * Copyright (C) 2017 Mandelkow
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Affero General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU Affero General Public License for more details.
 *
 * You should have received a copy of the GNU Affero General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

/**
 * @var PDR_FILE_SYSTEM_APPLICATION_PATH The full path of the application root
    as determined by the position of the default.php
 */
define('PDR_FILE_SYSTEM_APPLICATION_PATH', __DIR__ . '/');
/**
 * @var PDR_HTTP_SERVER_APPLICATION_PATH The relative path of the application
    root on the web server.
 */
$folder_tree_depth_in_chars = strlen(substr(getcwd(), strlen(__DIR__)));
$root_folder = substr(dirname($_SERVER["SCRIPT_NAME"]), 0, strlen(dirname(
   $_SERVER["SCRIPT_NAME"])) - $folder_tree_depth_in_chars) . "/";
define('PDR_HTTP_SERVER_APPLICATION_PATH', $root_folder);
//TODO: This does not work, if the location is a symbolic link.
/**
 * @var PDR_ONE_DAY_IN_SECONDS The amount of seconds in one day.
 */
define('PDR_ONE_DAY_IN_SECONDS', 24 * 60 * 60);

/*
 * Define an autoloader:
```

```
38  */
39 spl_autoload_register(function ($class_name) {
40     include_once PDR_FILE_SYSTEM_APPLICATION_PATH . 'src/php/classes/class.' .
       $class_name . '.php';
41 });
42
43
44 if (!file_exists(PDR_FILE_SYSTEM_APPLICATION_PATH . '/config/config.php')) {
45     header("Location: " . PDR_HTTP_SERVER_APPLICATION_PATH . "src/php/pages/
       install_page_intro.php");
46     die("The application does not seem to be installed. Please see the <a href='
       " . PDR_HTTP_SERVER_APPLICATION_PATH . "src/php/pages/install_page_intro.php
       '>installation page</a>!");
47 } else {
48     $config = array();
49     global $config; //This has to be explicitly declared in order to work with
       PHPUnit
50
51     /*
52      * Load configuration parameters from the configuration file:
53      */
54     require_once PDR_FILE_SYSTEM_APPLICATION_PATH . "config/config.php";
55     /*
56      * Complement the configuration array with the default values for unset
       parameters:
57      */
58     foreach (configuration::$List_of_configuration_parameters as $key => $value)
       {
59         if (!isset($config[$key])) {
60             $config[$key] = $value;
61         }
62     }
63 }
64 /*
65  * Setup if errors should be reported to the user, if to log them, and where:
66  */
67 ini_set('display_errors', $config['display_errors']); //Display errors to the
       end user?
68 ini_set('log_errors', $config['log_errors']); //Log errors to file?
69 if ($config['log_errors'] or $config['display_errors']) {
70     /*
71      * Debug mode
72      */
73     //ini_set('zend.assertions', 1); //Assertions will be compiled AND executed.
74     ini_set('assert.exception', 1); //An exception will be thrown if an
       assertion fails.
75 } else {
76     //ini_set('zend.assertions', -1); //Assertions are not compiled.
77     ini_set('assert.exception', 0); //Only warnings would be shown if assertions
       were to be executed and failed.
78 }
79 ini_set('error_log', $config['error_log']); //Which file should errors be logged
       to?
80 error_reporting($config['error_reporting']); //Which errors should be reported?
81
82 /*
83  * We want some functions to be accessible in all scripts.
84  */
85 require_once PDR_FILE_SYSTEM_APPLICATION_PATH . "funktionen.php";
86
```

```php
/*
 * Setup the presentation of time values:
  //setlocale(LC_ALL, 'de_DE'); // Leider versteht die Datenbank dann nicht mehr , was die Kommata sollen.
 */
setlocale(LC_TIME, $config['LC_TIME']);
/*
 * Setup default timezone for date()
 */
date_default_timezone_set($config['timezone']);
/*
 * Setup the encoding for multibyte functions:
 * This is necessary for the usage of UTF-8 characters in functions like mb_substr()
 */
mb_internal_encoding($config['mb_internal_encoding']);
require_once PDR_FILE_SYSTEM_APPLICATION_PATH . 'src/php/localization.php';

/*
 * session management
 */
$session = new sessions;

/*
 * TODO: Get rid of this maybe?
 */
$List_of_branch_objects = branch::get_list_of_branch_objects();
/*
 * Guess the navigator (=browser) language from HTTP_ACCEPT_LANGUAGE:
 * This is used in the head.php
 */
$navigator_languages = preg_split('/[,;]/', filter_input(INPUT_SERVER, 'HTTP_ACCEPT_LANGUAGE', FILTER_SANITIZE_STRING));
$navigator_language = $navigator_languages[0]; //ignore the other options
```

../default.php

4.1.1 Directory structure

- 🗀`config/` Contains the configuration file config.php
- 🗀`css/` *obsolete*, use 🗀`src/css/` instead
- 🗀`docs/` This documentation and tools to build it
- 🗀`img/` Images used by the program
- 🗀`js/` *obsolete*, use 🗀`src/js/` instead
- 🗀`locale/` translation files for gettext, currently only german (de_DE)
- 🗀`src/` Most of the actual source code
  - 🗀`src/css` Style Sheets
  - 🗀`src/js` JavasScript
  - 🗀`src/php/` PHP: Hypertext Preprocessor
  - 🗀`src/php/classes/` Contains all the class files class.class_name.php

  - 🗀 `src/php/fragments/` parts of bigger pages, may be included via php require/include or loaded with JavaScript
  - 🗀 **`src/php/pages/`** This is the place for the single views, which the human user will use to look at the roster etc.
  - 🗀 `src/sql/` SQL Database Tables and Triggers

- 🗀 `tests/` Tests to find errors in the source code; This folder is listed in .gitignore. Only some files are part of the visible source.
- 🗀 `tmp/` A directory for temporary files. There is no automatic cleanup yet.
- 🗀 `upload/` The destination for uploaded content. Currently only specific *.PEP files produced by Awinta ASYS Smart are understood. Those files contain information about the amount of customers that have been served in the past.

4.1.2 Coding standards

This project aims to follow some coding style guide.

- Please avoid StudlyCaps and camelCase.
- Class constants MUST be declared in all upper case with underscore separators.
- Property names MUST be written in under_score.
- Plain variables and objects are written in all lowercase.
- Array names start with a singe Uppercase letter followed by lowercase characters.
- Method names MUST be written in under_score.
- Code MUST use 4 spaces for indenting, not tabs.
- Opening braces for classes and functions MUST go on the same line, and closing braces MUST go on the next line after the body.
- Opening braces for control structures SHOULD go on the same line, and closing braces MUST go on the next line after the body.

Disk space is not rare anymore. IDEs are helping with autocomplete. There is no need to abbreviate stuff. Please use long terms like user_email_notification_cache instead of usr_ml_ntfcn_ca or u_e_n_c.

4.1.3 The database

Currently there is only MySQL supported as a database management system (DBMS). The tables are:

- absence (illness, vacation and other kinds of absence)
- approval (saves for each day if the leader has officially authorized the roster)
- branch (information about the main pharmacy and possible branches)
- Dienstplan (the actual roster data; start, end, break)

- employees (employee data; employee_id, name, profession, abilities)
- employees_backup (a copy of the employees table with historical data archived)
- Feiertage (obsolete)
- principle_roster (the basic plan; start, end, lunch break; is used to suggest new rosters)
- maintenance (obsolete)
- Mandant (obsolete)
- Notdienst (dates of emergency services and the employees scheduled to them)
- opening_times_special (not used yet)
- opening_times (the opening and closing times of the branches, no GUI yet for editing)
- pdr_self (reflects the state of the application itself)
- pep_month_day (the relative amount of work on different days in the month)
- pep (the raw amount of work data, hashed to reduce the amount of deleted/ignored entries)
- pep_weekday_time (the amount of work at different times on different weekdays)
- pep_year_month (the relative amount of work on different months in the year)
- saturday_rotation (whose turn is it to work on which saturday?)
- saturday_rotation_teams (who belongs to which team for saturday's rotation?)
- Schulferien (not used yet)
- Stunden (overtime archive and balance)
- task_rotation (rotating assignment of employees to a task, e.g. compounding)
- user_email_notification_cache (not used yet)
- users_lost_password_token (tokens provided to change a forgotten password)
- users_privileges (the privileges of the user accounts)
- users (the user accounts; there has to be exacly one employee for every user account; there may be employees without user accounts)
- Wunschplan (obsolete)

A copy of all the table structures is stored in 🗀`src/sql/`. The directory also contains the file 🗀`src/sql/database_version_hash.php` which holds a SHA1 hash of all the structures returned by SHOW CREATE TABLE and SHOW CREATE TRIGGER after some modification. The hash is written by 🗀`tests/get-database-structure.php`, see the details in that file.

Maintenance of the database

There is a class *update_database*. This class holds a defined set of MySQL statements that alter the database structure from a known state in the past to the current state.

This class is not well tested. It might work. It might as well destroy the whole database.

The class *update_database* is called on every login of a user. It then decides on its own, if any actions have to be taken. In order to decide, the hash stored in the file `database_version_hash.php` is compared to the hash stored in the database table pdr_self » pdr_database_version_hash.

Auto healing tables The class *database_wrapper* has a function *create_table_from_template()* that is able to create missing tables from the structure information given in `src/sql/`. It is called if any PDO database query throws an exception with the code 42S02 and the MySQL error 1146.

4.1.4 Classes

The classes are stored in the folder `src/php/classes/` or `src/php/3rdparty/`. The autoloader will only load classes from `src/php/classes/class.$class_name.php`, every other class has to be manually included.

user

The user class represents an employee, who has registered a user account in PDR.

user_input

user_input::get_variable_from_any_input This function reads user input from POST, GET or COOKIE in that order. If the requested information is found in one of the sources, then the others are ignored. For security reasons all information is filtered. By default FILTER_SANITIZE_STRING is used. Any other filter can be given as the second parameter. If no information is found in any of the sources, then a default value (the third parameter) will be returned.

escape_sql_value foo

convert_post_empty_to_php_null foo

principle_employee_roster_write_user_input_to_database foo

principle_roster_write_user_input_to_database foo

get_Roster_from_POST_secure foo

remove_changed_entries_from_database foo

remove_changed_entries_from_database_principle_roster foo

insert_changed_entries_into_database_principle_roster foo

insert_new_approval_into_database foo

old_write_approval_to_database foo

get_changed_roster_employee_id_list foo

get_deleted_roster_employee_id_list foo

get_inserted_roster_employee_id_list foo

roster_write_user_input_to_database foo

4.1.5 Web Interface

4.1.6 Calendar API

The script does not discriminate between information sent by POST, GET or COOKIE. See user_input::get_variable_from_any_input for details. The parameter date_string acepts any string that can be interpreted by DateTime See https://secure.php.net/manual/en/datetime.formats.date.php for details.

4.2 Documentation

This documentation about a programm, app or script is a stub. You can help this project by expanding it. Seriously, if there is something that does not explain itself enough, just send me an email or contact me at GitHub!

4.3 Testing

4.4 Bug tracker

Bugs and Issues are tracked at GitHub https://github.com/MaMaKow/dienstplan-apotheke/issues

4.5 Translation

Translations are handled with gettext().

See this article about po4a about the translation of this document: https://maltris.org/mehrsprachigkeit-fur-fast-alles-po4a-7317.html

4.5.1 Internationalization

Different countries have different laws regarding pharmacies and employment. They also have different holidays.