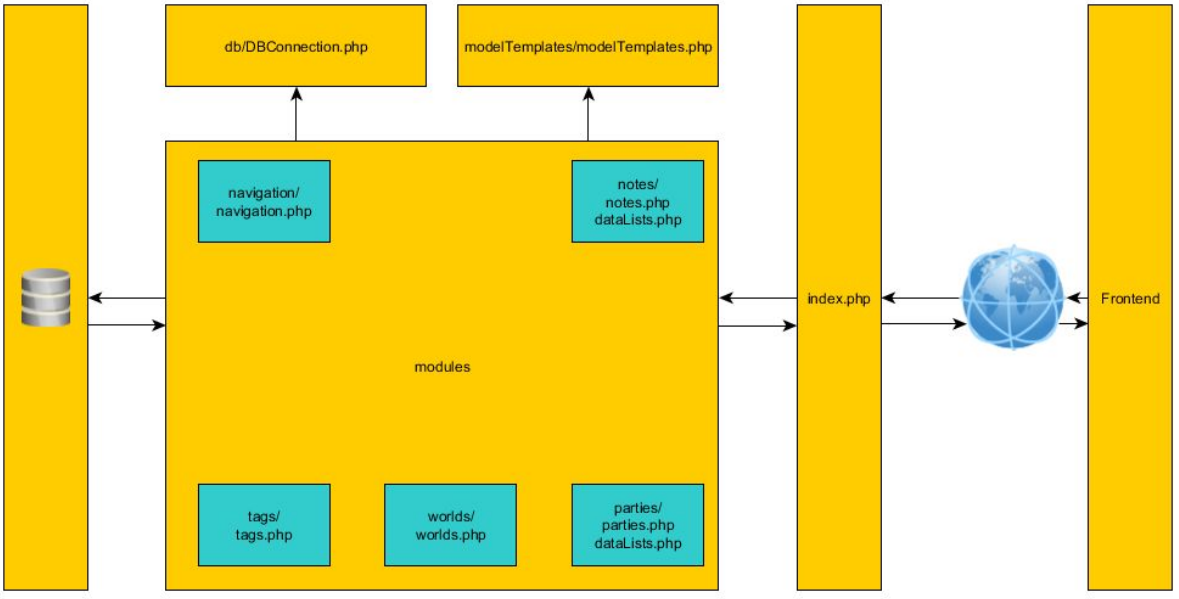
PnPBoardBackend

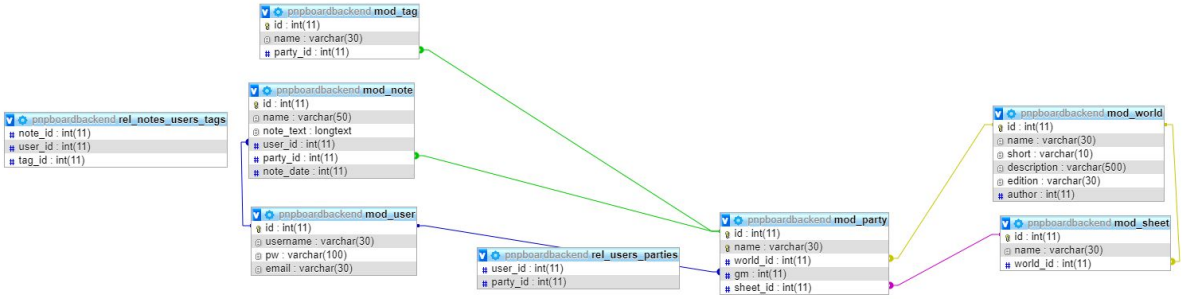
Installation:

1. downloade und installiere Lamppstack (je nach Betriebssystem: Xampp, Lampp, Wampp) mit der PHP Version: 7.2.3 (https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.2.3/)
2. kopiere das Projektverzeichnis in das htdocs Verzeichnis
3. starte über das Xampp Control-Panel den Apache und MySQL Server
4. öffne den Browser und gehe auf localhost/phpmyadmin
5. lege einen neuen Nutzer an
6. importiere (im phpMyAdmin) die gewühnschte Datenbank
    a. mit Testdaten: pnpboardbackend_testDB.sql
    b. ohne Testdaten: pnpboardbackend_rel_setup.sql
7. öffne DBConnection.php Datei und trage hier die, unter 5. angelegten, Nutzerdaten ein

Übersicht:



ER-Modell:



Datenaustausch:

Das Backend bekommt seine Daten als HTTP-Parameter. Die Daten die an das Frontend zurück gehen werden wiederum als JSON an das Frontend geliefert.

URL-Aufbau:

Bsp: localhost/pnpboardbackend?module=world&action=load&data[id]=1

Hier findet ein 3 teiliger Aufbau statt:

- **module:** gibt das an zusteuernde Modul an
    - Parameter:
        - navigation
        - worlds
        - parties
        - notes
        - tags
- **action:** gibt innerhalb der Module die anzusteuernde Funktion an
    - mögliche Parameter:
        - index → holt Daten für die index Views im Frontend
        - load → holt Daten für ein spezielles Element im Frontend
        - edit → updated Daten in der Datenbank
        - add → fügt neues Datenset in der Datenbank hinzu
        - delete → löscht ein spezielles Datenset in der Datenbank
- **data:** übergibt die, zur Verarbeitung, benötigten Daten. Ist ein Array, welches das zum Modul passende Modell abbildet.

Modulstruktur:

- <Modulname>.php ist die grundlegend wichtige Datei zur Datenverarbeitung eines Moduls.

- dataLists.php ist für zusätzlich Datensets verantwortlich die zur Verarbeitung von Daten im Frontend gebraucht werden.

Klassen:

Router:

Diese Klasse nimmt die Daten/Befehle die als HTTP-Request rein kommen entgegen und verteilt sie auf die entsprechenden Module.
Zusätzlich gibt es die geladenen Daten als JSON zurück.

Funktionen:

- __construct() : void
    - Der Konstruktor splittet den Request in seine Parameter auf und gibt diese an das jeweils entsprechende Modul weiter.
- returnData() : void
    - Diese Funktion wandelt die Datenarrays in JSON um und gibt sie an das Frontend zurück.

ModelTemplates:

Diese Klasse ist dazu da Daten, die von der Datenbank kommen, in das passende Format der jeweils zugehörigen Frontendmodels zu formatieren.

Fukntionen:

- userTempl(Array $usrArr) : Array
    - gibt ein Array in Form des Users zurück
- worldTempl(Array $worldArr) : Array
    - gibt ein Array in Form der World zurück
- partyTempl(Array $partyArr) : Array
    - gibt ein Array in Form der Party zurück
- sheetTempl(Array $sheetArr) : Array
    - gibt ein Array in Form eines Sheets zurück
- noteTempl(Array $noteArr) : Array
    - gibt ein Array in Form einer Note (Notiz) zurück
- tagTempl(Array $tagArr) : Array
    - gibt ein Array in Form eines Tags zurück

DBConnection:

Diese Klasse stellt die Datenbankverbindung her und stellt einige Metafunktion zur Arbeit/Verarbeitung der Daten von der Datenbank parat.

Funktionen:

- __construct()
    - Stellt die Datenbankverbindung her
- mysqliToData($mysqli) : Array
    - Wandelt ein mysqli Objekt in ein Datenarray um und gibt dieses zurück
    - Parameter: $mysqli = mysqli Objekt
- convertKey(Array &$arr) : void
    - Nimmt die Referenz eines Datenarray entgegen und sucht in den Array Indizes nach Wörtern die mit _ geschrieben sind und wandelt sie in Camelcase um

Modulklassen allgemein:

- dbAction() : Array
    - initialisiert die Datenbank
    - leitet anhand der vorgegebenen Action an die entsprechende Funktion weiter
    - gibt Datenarray zurück
- getIndex(Array &$data) : Array **bzw.** getIndex() : Array
    - Holt die Daten die für die Indexview / Listenansicht gebraucht wird
    - Param falls vorhanden:
        - data[id]=<id>
- loadData(Array &$data) : Array
    - Holt die Daten die dem ausgewählten Model gehören und gibt diese zurück
    - Param:
        - data[id]=<id>
- addData(&$data) : bool
    - Fügt einen Datensatz in der Datenbank in die zum Model zugehörige Tabelle/n ein
    - Param
        - Array Version des Frontendmodels
- editData(Array &$data) : bool
    - Updated den über das Array angegebenen Datensatz auf der Datenbank
    - Param
        - Array Version des Frontendmodels
- deleteData(&$data) : bool
    - Löscht den über das Array angegebenen Datensatz in der Datenbank
    - Param:
        - data[id]=<id>

DataLists allgemein:

- dbAction() : Array
    - initialisiert die Datenbank

    - leitet anhand der vorgegebenen Action an die entsprechende Funktion weiter
    - gibt Datenarray zurück
- get<Modell>List() : Array
    - Holt die jeweils zum angegebenen Model zugehörigen Datensets gibt diese zurück