**Имена:**  *Виктор Велинов Русев* **фн:**  *81644***Начална година:** *2021* **Програма:** *бакалавър, (КН)* **Курс:** *4*   
**Тема: Генератор на файл VSCode tasks.json  
Дата: 2022-02-15 Предмет: w17ref\_KN \_final имейл: viktor@rusev.dev**

**преподавател:** доц. д-р Милен Петров

# ТЕМА: Генератор на файл VSCode tasks.json

## 0. GitHub Repository

<https://github.com/vikrusev/fmi-final-web-project>

## 1. Условие

## Различните текстови редактори и среди за разработка имат различни аспекти, които могат да се настройват. Да се направи система за управление на настройки на задачите в редактора VSCode ( tasks.json ) и настройка на бързи клавиши за изпълнение ( shortcuts ). Трансформиране от json към shell ( под windows и linux ).

## 2. Въведение

Този проект позволява интерактивно създаване на файла **tasks.json** за **VSCode.**

Представлява логин форма *( index.php )*, форма за избиране на опции за файла *( pages/form.php )* , представяне на текущия избор *( pages/present.php )*, запазване на генерираната **json** конфигурация в клипборда на потребителя и история на конфигурациите *( pages/history.php )*.

Има кратка аутентикация за отделните страници, напр. потребител не може да отвори *pages/form.php*, ако не е влезнал в системата.

## 3. Теория

В класа *schemas/task\_schema.php* се описват различните опции, които да бъдат част от конфигурацията, от която потребителят може да създаде **tasks.json.**

Структурата е пригодена така, че лесно да се добавя или маха дадено поле.

## 4. Използвани технологии

*HTML5, CSS3, Vanilla JS*

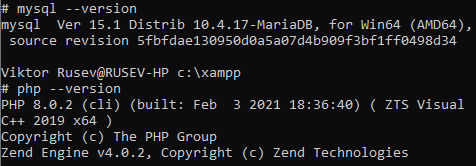
*XAMPP Control Panel v.3.2.4*

*PHP 8.0.2 (cli) (built: Feb 3 2021 18:36:40) ( ZTS Visual C++ 2019 x64 ) Copyright (c) The PHP Group Zend Engine v4.0.2, Copyright (c) Zend Technologies*

*MySQL mysql Ver 15.1 Distrib 10.4.17-MariaDB, for Win64 (AMD64), source revision 5fbfdae130950d0a5a07d4b909f3bf1ff0498d34*

База – *MariaDB*

Сървър – *Apache*



## 5. Инсталация и настройки

Проектът използва *Phinx Migrations*. За начална настройка на проекта *( в случая само на базата )*, потребителят е необходимо да изпълни следните стъпки:

* cd src *(– основна работна директория)*
* във файла phinx.php – да опише коректно данните на базата и акаунтът, с който ще се свързва към базата, съответно за една от трите налични среди – production, development и testing
* composer install *(– инсталиране на пакети от composer.lock файла)*
* vendor/bin/phinx migrate -e production *(– стартиране на миграциите)*

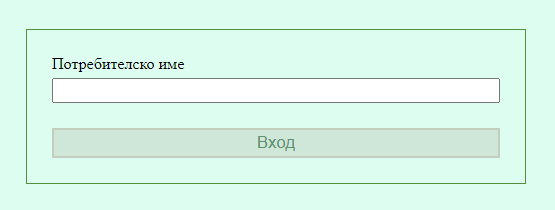
*Apache* сервира директорията */src*, в която се намира *index.php* – логин формата.

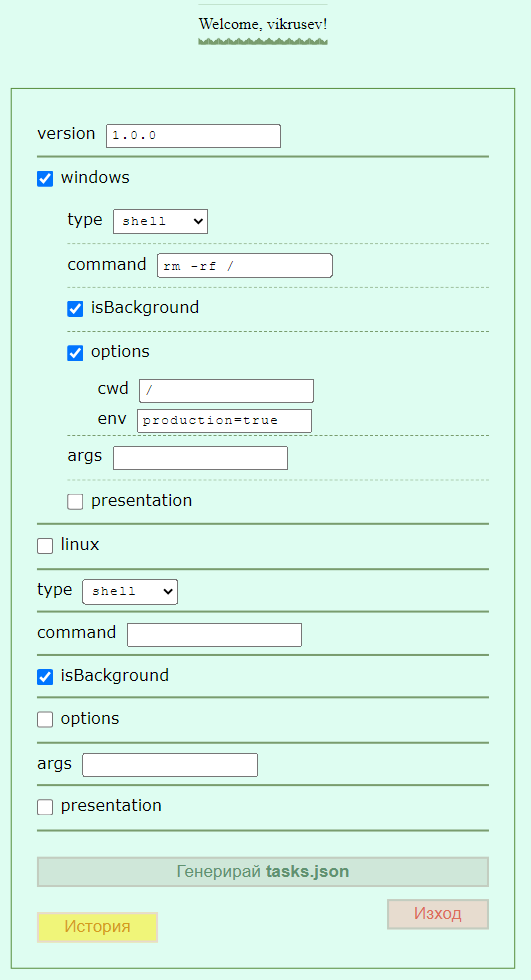
Базата автоматично се настройва. При първа *POST* заявка към кой да е *endpoint* от */server*, с изключение *logout.php* се подсигурява, че базата *fmi\_web\_project* съществува на *localhost*, както и необходимите таблици за проекта с правилни колони – *users* и *histories.*

## 6. Кратко ръководство на потребителя

На *index* страницата има единствено поле за потребителско име. За по-лесна реализация и работа по съществената част от проекта, валидация на потребителско име с парола, забравена парола и т.н. не са имплементирани.

Потребителят въвежда името си – ако не съществува в таблицата *users* – той бива добавен.



След успешен вход се отваря */pages/form.php*. Тази форма представлява интерактивно „създаване“ на **tasks.json.** Първото поле представлява името на конфигурацията, което ще се показва в историята на конфигурации, която се достъпва чрез бутона „История“.

Бутонът „Генерирай **tasks.json**” прави заявка към сървъра, запазва конфигурацията в таблицата *histories*, свързва я с текущо влезналия потребител и я визуализира на екрана в *JSON\_PRETTY* формат. Потребителят може да я копира в клипборда си натискайки бутона „Запазване в клипборда“ или да се върне към формата с бутона „Назад“.



Бутонът „История“ отваря страницата */pages/history.php*, на която се показват историите на конфигурации на съответния потребител. Всяка конфигурация може да бъде запазена в клипборда с бутона *„Запазване в клипборда“* под съответната конфигурация.

Потребителят може да излезе от акаунта си чрез бутона „Изход“ в */form.php.*

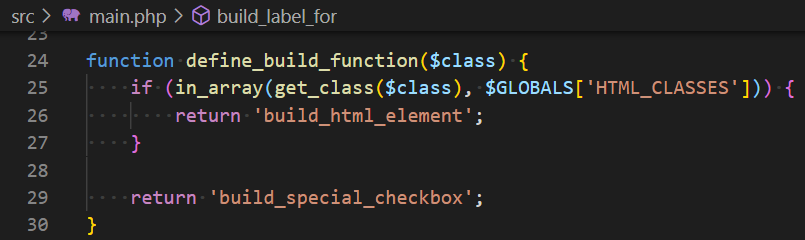
## 7. Описание на програмния код

### Схемата

Схемата, която се използва за генериране на формата в */pages/form.php* е описание на атрибути на клас *TaskConfiguration*. Атрибутите биват обходени един по един и с помощта на *PHP* рефлекция се проверява какъв клас са.

Налични са следните класове:

* **TaskConfiguration** extends **BaseTaskConfiguration**
* **CommandOption**
* **PresentationOptions**
* **HTML** класове *(HTML\_)*
  + **HTML\_Field**
  + **HTML\_Text** extends **HTML\_Field**
  + **HTML\_Checkbox** extends **HTML\_Field**
  + **HTML\_Select** extends **HTML\_Field**

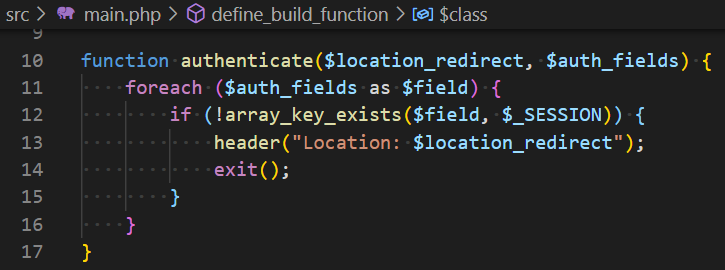
Всеки един атрибут на основния *TaskConfiguration* бива рекурсивно проверяван за класа му и в случай че е част от *HTML\_* класовете – той елементът бива генериран като съответния *DOM* елемент ( взет съответно от името на класа ). Той може да приема стойности по подразбиране, дали даденото поле е задължително и дали е селектирано ( чекбокс ).

### Страниците

Всяка страница е защитена от достъпване без необходимите потребителски данни за тази цел.

Напр. потребителят не може да отвори */pages/form.php*, */pages/present.php*, */pages/histories.php*, ако няма създадена сесия за него ( бива пренасочен към *index.php* ).

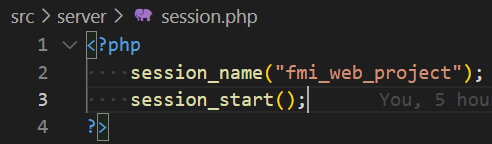
Това е постигнато с помощта на метода *authenticate(pathToRedirect, [<what data is required to open the page>])*

**

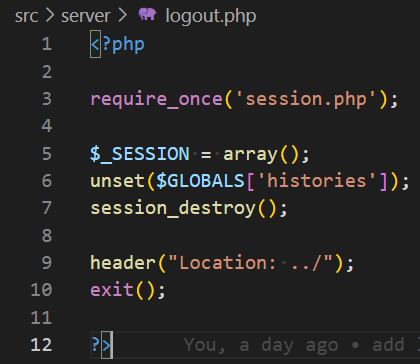
Ако потребителят е влезнал в системата, той не може да отиде на *index.php* страницата, ако не унищожи сесията си, т.е. да затвори таба, браузъра си или натисне бутона „Изход“.

### Сървър

Потребителят не може да изпълни заявка към сървъра */server/generate\_file.php* и */server/get\_history.php*, ако няма създадена сесия за него ( бива пренасочен към *index.php* ).

За потребителска сесия се използва *PHP* сесия – *session\_start()* като предварително се задава име на сесията чрез *session\_name(<name>) –* намира се в */server/session.php*

При излизане от системата данните за сесията се зануляват и сесията с *PHP* се унищожава. Също така се изтрива и глобалния масив *$GLOBALS[‘histories’]*, който се пази в *$GLOBALS* с цел улеснение на заданието.



При неуспешна връзка към базата – приложението „спира“ с *exit()*.

## 8. Приноси на студента, ограничения и възможности за бъдещо разширение

В бъдеще проектът може да се разшири по следните направления:

* Увеличаване наличния набор от полета в конфигурацията *( /schemas/task\_schema.php )*
* Добавяне темплейт за грешки (HTTP Code 5xx)
* проверка за съществуваща конфигурация с дадено име
* добавяне оторизация за акаунтите (защита с парола)
* показване на стилизирани грешки на страницата

## 9. Какво научих

*PHP* има *Reflection API.*

В *PHP* е възможно да се извиква функция като се използва името й в стринг-формат.

В HTML, според *W3* спецификацията, елементът *<button>* изпраща формата към указания *endpoint.* За да се пренебрегне това държание трябва изрично да се подаде атрибутът *type=”button”.*

Има „бъг“ с *<input type=”checkbox” />.* Чекбоксът е „чекнат“ след като се натисне бутон „назад”, въпреки че няма „checked” атрибут.

С помощта на *CSS* може да се създаде зиг-заг бордър.

## 10. Използвани източници

**Phinx Migrations**

<https://phinx.org/>

**Button Submit**

[https://www.w3.org/TR/2011/WD-html5-20110525/the-button-element.html#the-button-element](https://www.w3.org/TR/2011/WD-html5-20110525/the-button-element.html%23the-button-element)

<https://stackoverflow.com/a/10836076/13213781>

**Checkbox Bug**

<https://stackoverflow.com/questions/33731067/checkbox-still-checked-hitting-the-back-button-but-it-has-no-checked-attribut>

Предал (подпис): ………………………….

/*фн: 81644, имена: Виктор Велинов Русев, спец. КН, група 7*/

Приел (подпис): ………………………….

/доц. *Милен Петров*/