POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Podstawy teleinformatyki

Dokumentacja projektu

TABLICA INFORMACYJNA

Karol Rosiak

Kamil Rogowski

Damian Reczulski

14 czerwca 2019

Spis treści

Wstęp

Założeniem projektu było stworzenie tablicy informacyjnej bazującej na urządzeniu SBC Raspberry Pi. Wyświetlać miała ona elementy multimedialne takie jak zdjęcia, filmy wideo, strony internetowe (pobrane z adresu lub pliku html) oraz obraz z kamer IP. Na dole ekranu miał pojawiać się horyzontalny pasek informacyjny z aktualnym czasem wraz danymi pogodowymi. Dane multimedialne oraz napisy wyświetlane na pasku mogły być dodawane przez dedykowany panel administracyjny na stronie WWW. W panelu miał także znajdować się podgląd aktualnie wyświetlanych elementów na tablicy. Wszelkie zasoby są wyświetlane na zewnętrznym ekranie monitora w rozdzielczości 800x600, który pełni rolę tablicy informacyjnej. Informacje mogą być również wyświetlane na rzutniku multimedialnym, wtedy obowiązuje rozdzielczość ekranu 1024x768. Ustawienie odpowiedniej rozdzielczości jest determinowane przez konfigurację Raspberry Pi i stosowną ilością klatek na sekundę (FPS)

Temat projektu został wybrany ze względu na możliwość wykorzystania urządzenia typu Single-board computer. W przeszłości mieliśmy już styczność z programowaniem na takich urządzeniach, a jeden członek z naszego zespołu szczególnie pasjonuje się tego typu urządzeniami i w wolnym czasie w przeszłości skonfigurował system nawigacji GPS (z wykorzystaniem dodatkowych komponentów).

Podział prac

Tabela 1. Zadania poszczególnych członków zespołu

	Część I (20.03)	Część 2 (3.04)	Część 3 (17.04)	Część 4 (15.05)	Część 5 (29.05 i 12.06)
Damian Reczulski	Wstępna konfiguracja Raspberry Pi	Galeria zdjęć	Wyświetlanie wideo	Wyświetlanie zdalnych stron WWW i dokumentów HTML	Streamowanie obrazu z kamer
Kamil Rogowski	Panel logowania z funkcją haszowania hasła	Harmonogram - kolejność występowania elementów	Pobieranie zasobów i tworzenie kolejek	Wyświetlanie aktualnych danych pogodowych	Pasek informacyjny z ustawionym czasem wyświetlania
Karol Rosiak	Protokół przesyłania plików	Interfejs graficzny panelu administratora	Funkcjonalność panelu WWW	Podgląd przez panel WWW obrazu (screenshot)	Podgląd przez panel WWW wyświetlanego obrazu (wideo)

Funkcjonalności aplikacji

- wyświetlanie dokumentów HTML,
- wyświetlanie obrazów w formatach JPG, PNG, BMP, GIF (statyczny),
- wyświetlanie filmów (bez dźwięku),
- wyświetlanie zdalnych stron WWW,
- streamowanie obrazu z kamery,
- określanie daty pojawiania się zasobów (np. od 1 lipca do 3 lipca) oraz czasu ich wyświetlania (np. 40 sekund),
- występowanie paska informacyjnego na dole ekranu z ustawioną wiadomością tekstową, z możliwością regulacji prędkości przesuwania się tekstu i czasem wyświetlania do dnia, godziny, minuty i sekundy,
- występowanie widgetu odnoszącego się do aktualnej pogody, pokazującego temperaturę, ikonę odzwierciedlającą obecny stan pogody (np. słońce, chmury), ciśnienie w hektopaskalach [hPa] oraz aktualną godzinę,

- strona internetowa panelu administracyjnego
    - panel logowania z metodą haszowania hasła,
    - definiowanie czasu i daty wyświetlania elementów,
    - usuwanie obiektów z harmonogramu,
    - zarządzanie zawartością i czasem wyświetlania paska informacyjnego,
    - podgląd aktualnie wyświetlanej treści na tablicy informacyjnej,
    - dodawanie nowych obiektów zasobów (np. obrazy, wideo) do harmonogramu.

Użyty sprzęt oraz technologie

Sprzęt

Raspberry pi

Model użyty w projekcie to Raspberry Pi 3 B+. Jest to aktualnie model z najlepszymi parametrami sprzętowymi, co pozwoliło na zaimplementowanie projektu w najskuteczniejszej możliwej postaci. System operacyjny wykorzystywany na urządzeniu to Ubuntu MATE w wersji 18.04.2 (64-bit). Płytka jest wyposażona w cztery gniazda USB, gniazdo HDMI, złącze na kartę microSD, piny GPIO i gniazdo microUSB do zasilania układu. Zawiera procesor Broadcom SoC 700 MHz oraz 512 MB pamięci RAM i .Pozwala na komunikację UART i I2C.

Oprogramowanie

Apache2

Apache jest bezpłatnym, udostępnionym na zasadach Open Source serwerem WWW. Możliwości Apache pozwalają stosować go jako serwer obsługujący dowolne rodzaje witryn internetowych - począwszy od statycznych stron WWW, czy bardziej rozbudowanych witryn korzystających z dynamicznego generowania plików HTML, a skończywszy na ogromnych systemach bazodanowych w rodzaju hurtowni danych. Serwer apache został użyty z powodu dobrej kompatybilności z językiem PHP i małego zapotrzebowania zasobów sprzętowych.

MongoDB

Otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się dużą skalowalnością, wydajnością oraz brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Użyliśmy tego serwera baz danych z powodu braku relacji między danymi przechowywanymi w bazie danych. Wszystkie wpisy do bazy danych można było zapisać w formie dokumentów nie połączonych ze sobą w żaden sposób.

Nginx

Nginx jest darmowym serwerem HTTP open-source. Znany jest ze swojej wysokiej wydajności, stabilności, wielu bogatych funkcji oraz łatwej konfiguracji i małego zapotrzebowania zasobów. Użyto go jako serwer RTMP (ang. Real Time Messaging Protocol) i HLS (ang. HTTP Live Streaming) udostępniający podgląd w panelu administratora. Wybrano go z powodu dostępu obydwu modułów potrzebnych do streamowania obrazu oraz z powodu dobrze opisanego procesu streamowania.

Git

Git to rozproszony system kontroli wersji. Stworzył go Linus Torvalds jako narzędzie wspomagające rozwój jądra Linux. Git stanowi wolne oprogramowanie i został opublikowany na licencji GNU GPL w wersji 2. W projekcie został użyty do udostępniania postępów na serwisie github oraz do łatwiejszego pobrania gotowego projektu na urządzenie raspberry pi.

Języki programowania

Python

Python to język programowania wysokiego poziomu ogólnego przeznaczenia, o rozbudowanym pakiecie bibliotek standardowych, którego ideą przewodnią jest czytelność i klarowność kodu źródłowego. Jego składnia cechuje się przejrzystością i zwięzłością. Wybór padł na język Python z powodu dostępu do przydatnych bibliotek (opencv, tkinter, pymongo) oraz z powodu prostoty programowania w języku.

PHP

Php to interpretowany skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym. Czynnikiem który wpłynął na wybór języka było małe zapotrzebowanie sprzętowe serwera PHP dzięki

czemu mógł on działać razem z tablicą informacyjną na tym samym sprzęcie. Dodatkowo nie ma problemu z działaniem na systemie Ubuntu.

JavaScript

JavaScript to skryptowy język programowania, stworzony przez firmę Netscape, najczęściej stosowany na stronach internetowych. W projekcie został użyty do wyświetlania podglądu w panelu administratora ponieważ tag <video> dodany w HTML nie obsługuje natywnie streamu HLS.

Instalacja i konfiguracja oprogramowań i środowisk

Ubuntu

Dystrybucją którą użyliśmy w projekcie jest Ubuntu 18.04 z środowiskiem graficznym MATE. Był to obraz dla architektury ARMv8. Jest to ważne ponieważ serwer MongoDB użyty w projekcie nie wspiera natywnie architektury armhf. W projekcie można użyć innej dystrybucji należy tylko pamiętać aby posiadała GUI i była zgodna z architekturą ARMv8. Przykładowy obraz Ubuntu można pobrać z oficjalnej strony Ubuntu Mate www.ubuntu-mate.org/download/ . Po pobraniu obrazu musimy go zapisać na karcie microSD. Wymagana jest karta o pojemności co najmniej 8 GB. Do zgrania obrazu na kartę można użyć programu open source balenaEtcher lub innych alternatyw. Po zapisaniu obrazu na dysk należy umieścić kartę w Raspberry PI i uruchomić sprzęt. Proces instalacji jest bardzo krótki. Podczas instalacji zostaniemy zapytani o wybór języka, strefy czasowej, nazwy użytkownika i hasło oraz o to czy chcemy połączyć się z siecią. Połączenie z siecią będzie wymagane do pobrania potrzebnych programów, do działania API pobierającego dane pogodowe oraz do wyświetlania stron internetowych w tablicy informacyjnej, dlatego zalecane jest skonfigurowanie połączenia od razu. Cały proces instalacji nie powinien zająć więcej niż 10 minut.

Apache HTTP Server

Do instalacji serwera apache2 z obsługą PHP użyjemy systemu zarządzania pakietami APT. Na początku zainstalujemy podstawowy serwer apache komendą:

```
sudo apt-get install apache2
```

Po pomyślnym zainstalowaniu doinstalujemy interpreter PHP oraz moduł umożliwiający apache z korzystania z niego

```
sudo apt-get install php libapache2-mod-php
```

Gdy pakiety zainstalują się pomyślnie należy sprawdzić czy serwer działa poprawnie. Po instalacji serwer apache powinien chodzić w tle, jeżeli jednak tak nie jest należy go uruchomić komendą

```
sudo /etc/init.d/apache2 start
```

lub

```
sudo service apache2 start
```

W kolejnym kroku napisze,y krótki kod php który pozwoli nam sprawdzić poprawność instalacji oraz wersję zainstalowanego pakietu PHP. Używając edytora tekstu, w tym przypadku nano, tworzymy nowy plik php w folderze głównym stron internetowych apache:

```
sudo nano /var/www/html/info.php
```

I wpisujemy poniższy kod:

```
<?php
phpinfo();
?>
```

Następnie restartujemy serwer apache:

```
sudo /etc/init.d/apache2 restart
```

lub

```
sudo service apache2 restart
```

Po zrestartowaniu serwera wchodzimy w przeglądarkę internetową i wpisujemy adres localhost/info.php. Jeżeli wszystko zostało zainstalowane pomyślnie powinna ukazać się

nam strona z podstawowymi informacjami o zainstalowanej wersji PHP. Warto zapamiętać jaka wersja PHP została zainstalowana ponieważ przyda się to w dalszych punktach.

Na tym nie kończy się jeszcze wstępna konfiguracja serwera. Obecnie nasz użytkownik nie ma praw zapisu do folderu /var/www/html/. Wcześniej użyliśmy praw super użytkownika, żeby to obejść ale nie jest to rozwiązaniem trwałym. Aby dać uprawnienia aktualnemu użytkownikowi to do folderów i plików używamy tych trzech komend:

```
sudo chown -R $USER:$USER /var/www/html/
sudo find /var/www/html -type d -exec chmod u+rwx {} +
sudo find /var/www/html -type f -exec chmod u+rw {} +
```

Pierwsza komenda nadaje prawa właściciela plików i folderów aktualnemu użytkownikowi. Zmienna $USER przechowuje nazwę aktualnego użytkownika. Parametr -R oznacza rekursywne wykonanie się komendy dla każdego pliku i folderu znajdującego się w folderze. Druga komenda nadaje uprawnienia odczytu i przeglądania folderów (parametr +x). Trzecia komenda nadaje uprawnienia odczytu i zapisu plików.

W naszym projekcie zaimplementowaliśmy formularz przesyłania plików przez panel administracyjny stworzony w języku PHP. Domyślne ustawienia serwera apache nie pozwalają na poprawne działanie tego formularza, dlatego należy dodać i zmienić parę elementów. Na początek dodamy dwa foldery. Jeden do przechowywania przesłanych plików tymczasowo do sprawdzenia, a drugi do przechowywania plików już zatwierdzonych przez serwer PHP i skrypt programu. Wpisujemy poniższe komendy:

```
mkdir /var/www/html/tmp
mkdir /var/www/html/uploads
```

Serwer apache nie używa konta aktualnie zalogowanego użytkownika ani konta roota, więc nie ma dostępu do tych folderów. Należy mu nadać odpowiednie uprawnienia:

```
sudo chown www-data.www-data /var/www/html/tmp
sudo chown www-data.www-data /var/www/html/uploads
```

www-data jest nazwą użytkownika którą używa apache.
Teraz musimy wskazać interpreterowi php gdzie ma przechowywać przesyłane przez nas pliki. Używając edytora tekstowego należy edytować plik sudo nano

/etc/php/7.x/apache2/php.ini gdzie x jest wersją php widoczną przez serwer apache (można to sprawdzić używając pliku info.php). Po otworzeniu pliku odszukujemy linijkę:

```
;sys_temp_dir = "/var/www/html/tmp"
```

Usuwamy z niej średnik, sprawiając że linijka nie będzie już traktowana jako komentarz. Po znaku równości dodajemy ścieżkę do folderu tj. "/var/www/html/tmp". Ostatecznie linijka ta powinna wyglądać tak

```
sys_temp_dir = "/var/www/html/tmp"
```

Ścieżkę, do której będą przenoszone pliki, które przejdą weryfikację tj. /var/www/html/uploads podajemy w samym kodzie php a nie w pliku php.ini.

Ostatnią rzeczą jaką należy zmodyfikować jest zwiększenie maksymalnej wielkości przesyłanego pliku. W tym samym pliku php.ini odszukujemy linie odnoszącą się do maksymalnego rozmiaru pliku:

```
upload_max_filesize = 2M
```

I zmieniamy jej wartość na większą np. 200M. Litera M oznacza że podana wielkość jest w megabajtach. Po edytowaniu pliku można zapisać plik.

Dodatkową rzeczą, którą można zrobić ale nie jest wymagana jest zmiana ilości maksymalnej ilości plików przesyłanych w jednym zapytaniu. Odpowiedzialna jest za to linijka:

```
max_file_uploads = 20
```

Można ustawić tą wartość na 1. Formularz na stronie pozwala tylko na wysłanie jednego pliku jednakże przez prostą edycję HTML można to zmienić i może to doprowadzić do zaśmiecenia urządzenia.

Serwer MongoDB

Do przechowywania informacji o mediach wyświetlanych w tablicy oraz napisach wyświetlanych na pasku używamy serwera MongoDB. Jednakże wersja dla architektury ARMv8 nie znajduje się w oficjalnym repozytorium Ubuntu. Należy je dodać ręcznie. Na początku dodajemy publiczny klucz dla systemu zarządzania pakietami.

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
9DA31620334BD75D9DCB49F368818C72E52529D4
```

Podczas pisania tej dokumentacji, pliki MongoDB nie znajdowały się w gałęzi dla najnowszej dystrybucji “Bionic” dlatego musimy pobrać ją z gałęzi poprzedniej dystrybucji “xenial”. Dodatkowo w najnowszej wersji lincurl4 zastąpiło libcurl3 wymagane do poprawnej instalacji serwera, dlatego musimy cofnąć się o jedną wersję. Proste apt-get wystarczy, aby to zrobić.

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu
xenial/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install libcurl3
sudo apt-get install -y mongodb-org
```

Po pobraniu plików, ponownie aktualizujemy system zarządzania pakietami aby wykrył nasze nowo pobrane pliki. Argument -y oznacza, że system nie będzie się nas odpytywać, czy chcemy, aby instalowany program zajął dodatkową ilość miejsca.

Dodatkową rzeczą, którą można zrobić, lecz nie jest konieczna jest dodanie przestrzeni wymiany. Przestrzeń wymiany nie powinna być potrzebna w normalnym użytkowaniu, jednakże może się zdarzyć sytuacja, że dodatkowa przestrzeń będzie potrzebna.

```
sudo dd if=/dev/zero of=/swapfile.img bs=1M count=2048
sudo mkswap /swapfile.img
sudo swapon /swapfile.img
cat /proc/swaps
```

Podany przykład dodaje 2GB pamięci wymiany.

Teraz możemy uruchomić nasz serwer i sprawdzić jego status.

```
sudo service mongod start
sudo service mongod status
```

Następnie musimy sprawić, żeby php widział nasz serwer MongoDB. Do instalacji sterownika MongoDB użyjemy PECL, czyli repozytorium dla dodatków do php. PECL nie występuje w oryginalnym repozytorium Ubuntu, więc musimy je dodać ręcznie. Dodatkowo musimy pobrać z repozytorium php7.0-dev, która umożliwi skompilowanie sterownika.

```
sudo apt-get install php7.0-dev
sudo pecl install mongodb
```

Po ściągnięciu sterownika musimy umieścić w plikach PHP informację o tym że chcemy używać tego sterownika. Używamy edytora tekstowego i tworzymy plik:

```
sudo nano /etc/php/7.x/mods-available/mongodb.ini
```

X jest tutaj numerem wersji php. W otwartym pliku wpisujemy:

```
extension=mongodb.so
```

Następnie tworzymy dowiązania do dwóch plików. Jeden odnosi się do folderu apache a drugi do samego interpretera php.

```
sudo ln -sv /etc/php/7.x/mods-available/mongodb.ini
/etc/php/7.x/apache2/conf.d/20-mongodb.ini

sudo ln -sv /etc/php/7.x/mods-available/mongodb.ini
/etc/php/7.x/cli/conf.d/20-mongodb.ini
```

X należy zamienić wersją php dostępną w apache. Komenda ln oznacza dowiązanie do pliku, a argument -sv oznacza, że wiązanie jest symboliczne, a nie twarde oraz to, że nazwy plików zostaną wyświetlone.

Po wykonaniu wszystkich komend należy zrestartować serwer:

```
sudo service apache2 restart
```

Composer

Composer jest narzędziem do zarządzania zależnościami w php. Dzięki niemu można instalować i aktualizować pakiety. My użyjemy go do dodania biblioteki MongoDB do naszego panelu administracyjnego. Na początku pobieramy programy które przydadzą się nam do pobrania composera:

```
sudo apt-get install curl php-mbstring git unzip
```

Następnie przechodzimy do folderu domowego użytkownika, ponieważ będzie to miejsce, w którym zainstalujemy composer. Najpierw pobieramy instalator composera z pomocą curla, a następnie instalujemy go przy pomocy php:

```
curl -sS https://getcomposer.org/installer -o composer-setup.php
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

Argument -sS oznacza, że curl jest uruchomiony w trybie cichym i wyświetli tylko błędy.
Po ściągnięciu i zainstalowaniu composera możemy sprawdzić poprawność instalacji poprzez wpisanie w terminalu komendy:

```
composer
```

Powinien wyświetlić nam się napis composer stworzony ze znaków ASCII oraz inne informacje. Świeżo po instalacji nie mamy uprawnień dostępu do folderu composera, co uniemożliwia instalowanie dodatków. Należy nadać stosowne uprawnienia do folderu:

```
sudo chown -R $USER:$USER $HOME/.composer
```

Argument -R oznacza, że wszystkie pliki i foldery znajdujące się w folderze dostaną uprawnienia w sposób rekursywny. Zmienna $USER przechowuje nazwę aktualnego użytkownika, a zmienna $HOME przechowuje ścieżkę katalogu domowego aktualnego użytkownika. Po nadaniu uprawnień do folderu możemy przystąpić do dodania biblioteki mongoDB do composera:

```
composer require "mongodb/mongodb=^1.0.0" --ignore-platform-reqs
```

Argument --ignore-platform-reqs ignoruje wymagania php, hhvm, lib-* and ext-* i wymusza instalację, pomimo tego, że teoretycznie lokalna maszyna nie spełnia wszystkich wymagań.

Python

Główna część tablicy informacyjnej została stworzona w języku Python 3. W wielu dystrybucjach Linuxa jest on instalowany automatycznie, jednakże gdyby tak nie było trzeba go pobrać z repozytorium:

```
sudo apt-get install python3.6
```

Oprócz tego należy pobrać instalator pakietów PIP. Należy uważać żeby zainstalować go dla wersji 3:

```
sudo apt-get install python3-pip
```

Przy pomocy pip3 możemy zainstalować poniższe biblioteki:

```
pip3 install imgkit
pip3 instal pymongo
```

Biblioteka pymongo służy do łączenia się z bazą danych. Biblioteka imgkit jest potrzebna do poprawnego działania biblioteki tkinter.

Przy pomocy apt-install możemy zainstalować poniższe biblioteki:

```
sudo apt-get install libopencv-dev python3-openc
sudo apt-get install python-tk
sudo apt-get install python3-pil python3-pil.imagetk
```

Biblioteka openCV służy do wyświetlania obrazów i filmów. Biblioteka tkinter posłużyła nam do stworzenia paska informacyjnego. Biblioteka pil użyta była do wczytania ikon do paska informacyjnego.

Git

Najłatwiejszym sposobem pobrania projektu tablicy informacyjnej będzie użycie gita. Na początek przechodzimy do folderu /var/www/html inicjalizujemy repozytorium. Następnie dodajemy zdalne repozytorium, wyodrębniamy pliki z repozytorium, a następnie pobieramy tylko pliki z folderu serwer:

```
cd /var/www/html
sudo git init
sudo git remote add tablica-informacyjna
https://github.com/karol-rosiak/Tablica-informacyjna
sudo git fetch tablica-informacyjna
sudo git checkout tablica-informacyjna/master -- server
```

Ostatnia komenda pobierze cały folder serwer. Jednakże nie chcemy plików mieć w folderze serwer tylko html:

```
sudo mv server/* .
sudo rmdir server
```

W kolejnym kroku należy usunąć domyślny plik index.html z folderu, jeżeli istnieje:

```
sudo rm index.html
```

Teraz po wpisaniu localhost w przeglądarkę powinien wyświetlić się panel logowania do panelu administracyjnego.

Po ściągnięciu części administracyjnej powtarzamy te kroki dla części wizualnej tablicy informacyjnej. Przechodzimy do innego folderu np ~/Desktop i pobieramy pliki. Tym razem nie musimy ich przenosić do innego folderu

```
cd /var/www/html
sudo git init
sudo git remote add tablica-informacyjna
https://github.com/karol-rosiak/Tablica-informacyjna
sudo git fetch tablica-informacyjna
sudo git checkout tablica-informacyjna/master -- Gallery
```

NGINX server

NGINX server został użyty do streamowania obrazu z raspberry pi do podglądu na stronie internetowej. Na początku pobieramy elementy niezbędne do instalacji, następnie pobieramy z githuba pliki instalacyjne serwera nginx i je wypakowujemy:

```
sudo apt-get install build-essential libpcre3 libpcre3-dev libssl-dev

sudo mkdir ~/build && cd ~/build

sudo git clone git://github.com/arut/nginx-rtmp-module.git

sudo wget http://nginx.org/download/nginx-1.10.2.tar.gz
sudo tar xzf nginx-1.10.2.tar.gz
```

Po wypakowaniu przechodzimy do nowo pobranego folderu, dodajemy moduł rtmp do konfiguracji serwera i kompilujemy program:

```
sudo ./configure --with-http_ssl_module --add-module=../nginx-rtmp-module
sudo make
sudo make install
```

W tym momencie w najnowszych dystrybucjach Ubuntu kompilacja nie uda się i zostaną wyświetlone błędy np.:

```
src/core/ngx_murmurhash.c:37:11: error: this statement may fall through
[-Werror=implicit-fallthrough
h = data[2] << 16;
~
src/core/ngx_murmurhash.c:38:5: note: here
case 2:
~
src/core/ngx_murmurhash.c:39:11: error: this statement may fall through
[-Werror=implicit-fallthrough=]
h = data[1] << 8;

src/core/ngx_murmurhash.c:40:5: note: here
case 1:
~ ==
```

Jest to spowodowane tym, że nowe kompilatory gcc implementują ostrzeżenie -Wimplicit-fallthrough. Sprawdza ono czy dany case w wyrażeniu switch nie “spada” do

kolejnego case (np. poprzez brak słowa break lub return). W kodzie programu nginx takie zachowanie “spadku” jest pożądane, i żeby dać o tym znać kompilatorowi należy po każdej linijce, w której występuje błąd dodać komentarz:

```
/* fallthrough */
```

Kompilator zostanie poinformowany, że dane zachowanie jest zamierzone, a nie przypadkowe. Po dodaniu komentarzy do każdej problematycznej linii możemy przejść ponownie do procesu kompilacji:

```
sudo make
sudo make install
```

Teraz kompilacja i instalacja powinny przejść pomyślnie. Możemy przetestować czy nasz serwer się uruchomi:

```
sudo /usr/local/nginx/sbin/nginx
```

Jeżeli nie wyświetliły się żadne informacje o błędach oznacza to, że serwer działa poprawnie. Nie jest to jeszcze koniec konfiguracji serwera dlatego na razie musimy go zatrzymać:

```
sudo /usr/local/nginx/sbin/nginx -s stop
```

Dalszą konfigurację zaczniemy od stworzenia folderów do streamowania i nadania uprawnień użytkownika do nich:

```
cd /
sudo mkdir HLS
sudo mkdir HLS/live
sudo mkdir video_recordings
sudo chmod -R 755 HLS
sudo chmod -R 777 video_recordings
```

HLS jest folderem, w którym przechowywane będą pliki streamów (można prowadzić kilka streamów na raz). Podfolder live jest folderem, w którym zlokalizowane będą streamowane

aktualnie pliki wideo oraz plik playlisty index.m3u8. Folder video_recordings jest opcjonalnym folderem do zapisywania archiwalnych transmisji. Kolejnym krokiem jest edycja konfiguracji. Na początku na wszelki wypadek tworzymy kopię oryginalnej konfiguracji, a następnie otwieramy plik konfiguracyjny edytorem tekstu:

```
cd /usr/local/nginx/conf
sudo cp nginx.conf nginx.conf.back
sudo nanonginx.conf
```

W edytorze tekstu zaznaczamy cały tekst i zamieniamy go poniższym kodem:

```
user  www-data;
worker_processes  1;

error_log  logs/error.log debug;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;

#pid        logs/nginx.pid;

events {
   worker_connections  1024;
}

http {
   include       mime.types;
   default_type  application/octet-stream;

   server {
      listen       2222;
         server_name localhost;

         location /live {
                        types {
                                application/vnd.apple.mpegurl m3u8;
                        }
                        alias /HLS/live;
                        # Disable cache
      add_header Cache-Control no-cache;

      # CORS setup
      add_header 'Access-Control-Allow-Origin' '*' always;
      add_header 'Access-Control-Expose-Headers' 'Content-Length';

      # allow CORS preflight requests
      if ($request_method = 'OPTIONS') {
```

```
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Max-Age' 1728000;
            add_header 'Content-Type' 'text/plain charset=UTF-8';
            add_header 'Content-Length' 0;
            return 204;
                    }
}
}
}

rtmp {
        server {
                listen 1935;
                allow play all;
                ping 30s;
                notify_method get;
                application live {
                        allow play all;
                        live on;
                        record off;
                        #record all;
                        #record_path /video_recordings;
                        #record_unique on;
                        hls on;
                        hls_nested on;
                        hls_path /HLS/live;
                        hls_fragment 5;
                        hls_playlist_length 8;
                        #deny play all;
                }
                application vod {
                        play /video_recordings;
                }
        }
}
```

Opis konfiguracji:

- user www-data - wskazuje serwerowi aby użyć użytkownika z którym współpracuje apache, dzięki czemu będzie miał dostęp do streamu,
- worker_processes 1 - ustala liczbę procesów,
- error_log logs/error.log debug - informacje o debugowaniu,
- #error_log logs/error.log notice - informacje o zdarzeniach normalnych, lecz wartych uwagi (tutaj wyłączone),
- #error_log logs/error.log info - informacyjne wiadomości które nie są wymagane do przeczytania, ale można je przeczytać (tutaj wyłączone),

- #pid logs/nginx.pid - pozwala na zmianę pliku w którym przechowywane jest PID głównego procesu serwera (tutaj wyłączone),
- worker_connections 1024 - ustala maksymalną jednoczesną ilość połączeń na jaką jeden worker może zezwolić,
- include mime.types - zezwala na użycie mime types,
- default_type application/octet-stream - ustawia typ mime na octet-stream,
- listen 2222 - ustawia port nasłuchiwania serwera HTTP na 2222. Jest to ważne ponieważ domyślny port 80 jest już zajęty przez apache,
- server_name localhost - adres serwera,
- location /live - określa który stream chcemy aktualnie konfigurować. /live oznacza ścieżkę dostępu do zasobu np. localhost:2222/live,
- types { application/vnd.apple.mpegurl m3u8;} - ustawia typ mime na odpowiadający naszej metodzie streamingu,
- alias /HLS/live - określa folder w którym znajdują się pliki do streamowania,
- add_header [...] - wszystkie dodane nagłówki mają za zadanie pozwolić na dostęp do streamu z innych źródeł tzn. spoza lokalnej maszyny,
- listen 1935 - ustawia port nasłuchiwania serwera rtmp,
- allow play all - zezwala na odtwarzanie streamu z każdego adresu,
- ping 30s - ustala po jakim czasie wysyłany jest pakiet kontrolny sprawdzający połączenie z klientem,
- notify_method get - ustala metodę HTPP do powiadomień,
- application live - wskazuje że będziemy teraz konfigurować konkretny stream w tym przypadku o nazwie live,
- live on - po uruchomieniu streamowanie jestod razu włączone,
- record off - wyłącza nagrywanie zapisu live do pliku,
- #record all; - włącza nagrywanie do pliku (tutaj nie używane),
- #record_path /video_recordings - ustala ścieżkę zapisu nagrań wideo (tutaj nie używane),
- #record_unique on; - gdy włączone, dodaje aktualne nagranie do istniejącego już pliku. Gdy wyłączone nadpisuje plik (tutaj nie używane),
- hls on - włącza HLS w aplikacji,
- hls_nested on - przy tej opcji, każdy stream ma swój własny folder,
- hls_path /HLS/live - ustala scieżkę do przechowywania aktualnych plików strumienia,
- hls_fragment 5 - ustala długość fragmentów HLS w sekundach,
- hls_playlist_length 8 - ustala długosc playlisty w sekundach,

- #deny play all - blokuje wszystkie połączenia (tutaj nie używane),
- application vod {play /video_recordings;} - definiuje aplikację do odtwarzania nagrań streamów.

FFmpeg

Do nagrywania ekranu i przekazywania obrazu do serwera streamingowego został użyty framework FFmpeg. Na początku musimy go pobrać poprzez apt-install:

```
sudo apt install ffmpeg
```

Program nie wymaga dalszej konfiguracji. Aby móc go przetestować musimy włączyć serwer rtmp:

```
sudo /usr/local/nginx/sbin/nginx
```

Po uruchomieniu serwera możemy zacząć test nagrywania i streamowania obrazu:

```
ffmpeg -f x11grab -s 800x600 -r 30 -i $DISPLAY -vcodec libx264 -threads 0 -crf 0 -preset ultrafast -f flv "rtmp://127.0.0.1:1935/live"
```

- -f x11grab - ustawienie wejścia , x11grab ustawia przechwytywanie pulpitu,
- -s 800x600 - wielkość, w tym przypadku rozdzielczość ekranu,
- -r 30 - ilość klatek na sekundę. Ustawione na 30,
- -i $DISPLAY- ustawienie wejścia dla x11grab. $DISPLAY przechowuje zmienną, która wskazuje na aktualny wyświetlacz (przykładowa wartość to :0.0),
- -vcodec - ustawienie kodeku wideo. W tym przypadku jest to libx264 ponieważ jest kodekiem MPEG4 wspieranym przy streamingu RTMP np. do Twitcha lub NGNIX Server,
- -threads 0 - ilość wątków jaka może być użyta do kodowania wideo. Przy opcji ustawionej na 0 może użyć wszystkich dostępnych wątków,
- -crf 0 - wskazuje kodekowi libx264, żeby przetwarzał obraz bezstratnie,
- -preset ultrafast - sugeruje kodekowi żeby przetwarzał obraz jak najszybciej może

- -f flv <adres> - ustawienie formatu wyjścia jako flv ponieważ jest on obsługiwany przez RTMP. Na końcu podajemy gdzie dane mają być strumieniowane - w tym przypadku do naszego serwera RTMP,

Po odpaleniu nagrywania można przetestować stream. Można to sprawdzić np. używając programu VLC i podając mu jeden z dwóch adresów:

```
rtmp://192.168.43.71:2222/live
```

lub

```
http://192.168.43.71:2222/live/index.m3u8
```

Pierwszy link używa bezpośrednio protokołu RTMP a drugi link jest już streamem HLS, używającym HTTP.

Kody programów

Struktura bazy danych

W projekcie użyliśmy nierelacyjnej bazy danych MongoDB. W bazie tego typu tworzy się kolekcje niepowiązanych ze sobą dokumentów. Nasza baza danych posiada 4 kolekcje.

schedule

```
{
	"_id" : ObjectId,
	"name" : string,
	"type" : string,
	"duration" : int,
	"start" : string,
	"end" : string
}
```

Dokument schedule jest odpowiedzialny za przechowywanie danych o przesłanych plikach na serwer, linkach stron do wyświetlenia oraz adresach IP kamer IP.
Opis pól:

- _id - pole identyfikujące dany dokument,

- name - nazwa pliku lub adres strony/kamery IP,
- type - typ zasobu. W projekcie wyróżniamy typy:
    - image - zdjęcie,
    - video - film wideo,
    - html - plik html,
    - url - link do strony,
    - webcam - link do kamery IP,
- duration - czas w sekundach jak długo element ma wyświetlać się na ekranie,
- start - data od której element ma pokazywać się na tablicy informacyjnej w formacie yyyy-MM-dd,
- end - data do której włącznie element ma pokazywać się na tablicy informacyjnej w formacie yyyy-MM-dd.

textSchedule

```
{
	"_id" : ObjectId,
	"text" : string,
	"start" : string,
	"end" : string
}
```

Dokument textSchedule ma podobną strukturę do dokumentu schedule. Przechowuje on napisy jakie mają pojawiać się na pasku informacyjnym.

Opis pól:

- _id - pole identyfikujące dany dokument,
- text - napis jaki ma zostać wyświetlony na pasku,
- start - data od której napis ma pokazywać się na tablicy informacyjnej w formacie yyyy-MM-dd,
- end - data do której włącznie napis ma pokazywać się na tablicy informacyjnej w formacie yyyy-MM-dd.

users

```
{
	"_id" : ObjectId,
	"user" : string,
	"password" : string
}
```

Dokument users przechowuje dane logowania użytkownika do panelu administratora.

Opis pól:

- _id - pole identyfikujące dany dokument,
- user - login użytkownika,
- password - zahaszowane hasło użytkownika. Do haszowania użyto algorytmu bcrypt.

weatherApi

```
{
	"_id" : ObjectId,
	"key" : string,
	"city" : string
}
```

Dokument weatherApi przechowuje informacje potrzebne do pobrania danych pogodowych z serwisu openWeatherMap.

Opis pól:

- _id - pole identyfikujące dany dokument
- key - przechowuje klucz do API serwisu OpenWeatherApi
- city - miasto o którym chcemy pobrać informacje pogodowe

Panel administratora

Panel administratora został napisany używając języka php i w małej części używając języka javascript. Frontend został stworzony używając HTML i CSS.

file.php

Plik ten przechowuje klasę odpowiedzialną za upload plików. Znajdują się w niej funkcje uploadu pliku oraz sprawdzania czy dany plik jest poprawny.

```
 function __construct(){
 if(isset($_FILES["fileToUpload"])){
   $this->target_file = $this->target_dir . basename($_FILES["fileToUpload"]["name"]);
   $this->target_file_name = basename($_FILES["fileToUpload"]["name"]);
   $this->fileType = strtolower(pathinfo($this->target_file,PATHINFO_EXTENSION));
 }
 }
```

Konstruktor klasy na początku sprawdza czy istnieje globalna zmienna $_FILES["fileToUpload"], w której znajdują się informacje o przesłanym pliku. Następnie zapisuje do zmiennych ścieżkę pliku. Podaje gdzie ma zostać zapisany, nazwę pliku oraz jego rozszerzenie.

```
private $imageType = array("jpg","jpeg","png","gif","bmp");
private $videoType = array("wmv","mp4","flv");
private $htmlType = array("html","xhtml");

/* … */

private function checkFileExtension(){
   switch($_POST["type"]){
     case "image":
       if(in_array($this->fileType,$this->imageType)) return true;
       else $this->uploadOk = 0;
     break;
     case "video":
       if(in_array($this->fileType,$this->videoType)) return true;
       else $uploadOk = 0;
     break;
     case "html":
       if(in_array($this->fileType,$this->htmlType)) return true;
         else $this->uploadOk = 0;
     break;
     default:
        $this->uploadOk = 0;
     break;
   }
   if($this->uploadOk == 0){
     $this->error = "Zły format pliku";
   }
 }
```

Funkcja sprawdzania, czy plik posiada rozszerzenie obsługiwane przez aplikacje. Obsługiwane rozszerzenia przechowywane są w tablicach zapisanych jako pola klasy. Na początku funkcji sprawdzany jest typ medium jaki został wybrany w formularzu. Następnie w

odpowiednim wyrażeniu case sprawdzane jest, czy wybrany typ medium odpowiada rozszerzeniu pliku. Jeżeli tak funkcja zwraca true, w przeciwnym wypadku ustawia wartość pola uploadOk na 0 i dodaje komentarz w zmiennej error.

```
function saveFile(){
  $this->checkIfExists();
  $this->checkFileExtension();

  if ($this->uploadOk == 1) {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $this->target_file)) {
       return true;
    } else {
      $this->error =  "Błąd podczas uploadowania pliku";
      return false;
    }
 }
 else{
   return false;
 }
}
```

Funkcja zapisująca plik. Przenosi ona plik z folderu tymczasowego do folderu podanego w kodzie. Na początku sprawdza czy plik na pewno istnieje i czy jego rozszerzenie jest zgodne z aplikacją. Następnie przenosi plik z folderu tymczasowego do folderu z uploadowanymi plikami. Jeżeli przy przenoszeniu nie napotkano żadnych problemów (jak na przykład brak uprawnień do folderu) funkcja zwraca true. W przeciwnym razie zwraca false i zmienia wartość zmiennej error na komentarz z błędem.

scheduleEntry.php

Plik ten przechowuje klasę odpowiedzialną za dodawanie i wyświetlanie informacji o elementach multimedialnych, które mają być wyświetlane na tablicy informacyjnej.

```
 private $connection;
 private $db;
 private $collection;

/* … */

 function __construct() {
   $this->connection = new MongoDB\Client("mongodb://localhost:27017");
   $this->db = $this->connection->tablica_informacyjna;
   $this->collection = $this->db->schedule;
 }
```

Konstruktor klasy. Na początku tworzy nowe połączenie z serwerem bazy danych MongoDB na domyślnym porcie 27017. Następnie wybiera bazę tablica_informacyjna. Na końcu wybiera kolekcję schedule. Jeżeli kolekcja nie istnieje, tworzy ją.

```
function checkDate(){
  if(isset($_POST["start"]) && isset($_POST["end"])){
    $dateStart = new DateTime($_POST["start"]);
    $dateEnd = new DateTime($_POST["end"]);
    if($dateEnd < $dateStart){
      return false;
    }
  }
  return true;
}
```

Funkcja sprawdzająca date. Sprawdza czy w formularzu wprowadzono datę startu i zakończenia wyświetlania elementu. Następnie zamienia dane ze stringa na obiekt typu DateTime. Jeżeli data zakończenia występuje przed datą rozpoczęcia zwraca false. W przeciwnym wypadku zwraca true.

```
function saveEnteryToDb($name,$type,$duration, $start,$end){
  $document = array( "name" => $name,"type" => $type, "duration" => $duration, "start"
=> $start,"end" => $end );
  $result = $this->collection->insertOne($document);
  if($result->getInsertedCount() > 0)
    return true;
  else
    return false;
}
```

Funkcja zapisująca dane do bazy danych. Jako argumenty przyjmuje po kolei nazwę (adres w przypadku strony internetowej i kamery IP), typ elementu, czas wyświetlania, czas rozpoczęcia wyświetlania i czas zakończenia wyświetlania. Następnie tworzona jest tablica asocjacyjna odpowiadająca strukturze dokumentu. Stworzona tablica użyta jest następnie przez funkcję biblioteki insertOne do zamieszczenia dokumentu w bazie danych. Jeżeli udało się poprawnie wstawić dokument funkcja zwróci true. W przeciwnym wypadku false.

```
function getEntriesByDate($date){
    $rangeQuery = array('start' => array( '$lte' => $date),'end' => array( '$gte' => $date));
    $result = $this->collection->find($rangeQuery);
    return $result;
}
```

Funkcja pobierająca dokumenty tylko z danego przedziału daty.Warunki filtrujące dokumenty należy umieścić w tablicy. $lte oznacza “mniejsze lub równe niż” a $gte oznacza “większe lub równe niż”. Po stworzeniu tablicy przekazywana jest ona do funkcji find() biblioteki MongoDB. Na końcu funkcja zwraca rezultat pobrany z bazy.

```
function deleteEntry($id){
  return $this->collection->deleteOne(array('_id' => new MongoDB\BSON\ObjectId($id)));
}
```

Funkcja usuwająca dokument. Jako argument podaje się identyfikator dokumentu. Następnie funkcja deleteOne() usuwa dokument jeżeli istnieje. Przy powodzeniu funkcja zwraca true, w przeciwnym wypadku false.

user.php

Plik przechowuje klasę odpowiedzialną za logowanie się użytkownika.

```
function getPassword($login){

    $result = $this->collection->findOne(array('user' => $login), array('password'));
    return $result["password"];
}
```

Funkcja pobierająca hash hasła z bazy danych. Jako argument funkcji podaje się login użytkownika. Do funkcji wyszukującej podajemy tablicę z nazwą użytkownika, w drugim argumencie wskazujemy, że interesuje nas kolumna z hasłem. Funkcja zwraca hash hasła lub pusty string w przypadku złych danych.

weatherApi.php

Plik zawierający klasę odpowiedzialną za zmianę informacji potrzebnych do działania API OpenWeatherApi - klucza API oraz miasta z którego chcemy się dowiedzieć pogody.

```
function saveEnteryToDb($key,$city){
   $document = array( "key" => $key,"city" => $city);

   $this->collection->deleteMany(array(),array('safe' => true));
   $result = $this->collection->insertOne($document);
   if($result->getInsertedCount() > 0)
     return true;
   else
     return false;
 }
```

Funkcja jako argument przyjmuje klucz API oraz nazwę miasta. Z podanych wartości tworzona jest tablica która zostanie użyta w funkcji dodającej dokument. W kolejnym kroku usuwane są wszystkie dokumenty będące w bazie, ponieważ aplikacja potrzebuje tylko jednego klucza i jednego miasta. Następnie dodawany jest dokument i jeżeli zostanie dodany prawidłowo funkcja zwróci true, w innym przypadku false.

login.php

Jest to strona wyświetlająca formularz logowania. Kod php sprawdza poprawność danych logowania.

```
session_start();
require_once($_SERVER["DOCUMENT_ROOT"] ."/objects/user.php");
if(!isset($_SESSION["zalogowany"])){
        if ($_SERVER['REQUEST_METHOD'] === 'POST') {
           if(isset($_POST["username"]) && isset($_POST["password"])){
                        $db = new Database();
                        $comparePassword = $db->getPassword($_POST["username"]);

if(password_verify($_POST["password"],$comparePassword)){
                                $_SESSION["zalogowany"] = $_POST["username"];
                                header('Location: index.php');
                                }else{
                                    $error="Niepoprawny login lub hasło";
                                }
                        }else{
                                $error = "Nie podano wszystkich daych";
                        }
        }
}
else{
        header('Location: index.php');
}

if(!empty($error)){
        echo "<div class='alert alert-danger' role='alert'> $error </div>";
}
```

Na początku sprawdzamy, czy użytkownik jest już zalogowany. Jeżeli jest przenosi go na stronę główną. Następnie sprawdzamy czy przyszły jakieś dane metodą POST i czy te dane to login oraz hasło. Jeżeli nic nie przyszło metodą POST strona wyświetla okno logowania. Gdy dane przyszły metodą POST pobierany jest hash hasła z bazy danych i metodą password_verify() jest porównywany z przesłanym hasłem. Jeżeli hasło przejdzie weryfikację, tworzona jest nowa zmienna sesyjna i użytkownik przenoszony jest do strony głównej. W przeciwnym wypadku wyświetlany jest komunikat o błędzie i użytkownik może wpisać dane ponownie.

schedule.php

Plik ten jest odpowiedzialny za wyświetlenie harmonogramu mediów z danego dnia w postaci tabeli.

```
if(!isset($_GET["date"])){
        $date = $dateToday = date("Y-m-d");
}else{
        $date = $_GET["date"];
}
$entries = $entryDb->getEntriesByDate($date);
/* … */
 <tbody>
            <?php
            $i=1;
            foreach($entries as $document) {
                  $name = $document["name"];
                  $type = $document["type"];
                   $start = $document["start"];
                   $end = $document["end"];
                   $duration = $document["duration"];

                echo "<tr>
                     <th scope='row'>$i</th>
                     <td>$name</td>
                     <td>$type</td>
                     <td>$start</td>
                     <td>$end</td>
                     <td>$duration</td>
                    </tr> ";
                    $i++;
            }
            ?>
          </tbody>
```

Na początku sprawdzane jest czy użytkownik przesłał datę metodą GET. Jeżeli nie, użyta zostaje aktualna data. Następnie z bazy danych pobrane są dokumenty zawierające wpisy mediów do wyświetlenia w podanej dacie. W dalszej części kodu wynik zapytania do bazy danych przypisywany jest do zmiennych dla ułatwienia zapisu. Następnie tworzone są kolejne wiersze tabeli z podanych danych.

upload.php

```
if(isset($_POST["submit"])) {
	$entry = new ScheduleEntry();
	$error = null;
	if(!$entry->checkDate()){
		$error = "Data zakończenia jest wcześniejsza od daty startu";
	}else{
		if(!empty($_FILES["fileToUpload"]["tmp_name"])){
			$file = new File();
			if($file->saveFile()){
if(!$entry->saveEnteryToDb($file->getName(),$_POST["type"],$_POST["duration"],
$_POST["start"],$_POST["end"])){
					$error = "Bład podczas dodawania do
harmonogramu";
				}
			}
			else{
				$error = $file->getError();
			}
		}
		else{
			if(!empty($_POST["link"]))$name = $_POST["link"];
			if(!empty($_POST["webcam"]))$name = $_POST["webcam"];

if(!$entry->saveEnteryToDb($name,$_POST["type"],$_POST["duration"],
$_POST["start"],$_POST["end"])){
				$error = "Bład podczas dodawania do harmonogramu";
			}
		}
	}
}
```

Plik ten zawiera stronę z formularzem dodawania mediów. Na początku sprawdzane jest czy przyszły jakieś dane metoda POST. Jeżeli nie wyświetlany jest formularz dodawania elementów do tablicy. Następnie sprawdzane jest czy w zapytaniu został przesłany plik czy nie. Jeżeli jest plikiem oprócz zapisu do bazy danych uruchamiana jest funkcja do zapisu pliku. W przypadku niepowodzenia którejś z funkcji wyświetlany jest komunikat o błędzie.

weatherApiTest.js

Jest to plik zawierający skrypt sprawdzający status OpenWeatherMap API. Dane pobierane są z formularza na stronie.

```
var getJSON = function(url, callback) {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.responseType = 'json';
    xhr.onload = function() {
      var status = xhr.status;
      if (status === 200) {
        callback(null, xhr.response);
      } else {
        callback(status, xhr.response);
      }
    };
    xhr.send();
};
```

Jest to funkcja wysyłająca zapytanie do API i odbierająca odpowiedź w formacie JSON. Na początku otwieramy połączenie metodą GET i ustawiamy postać odpowiedzi na JSON. Po otrzymaniu odpowiedzi sprawdzamy kod statusu. Jeżeli serwer zwrócił kod 200 oznacza to, że nie było żadnych błędów. W innym przypadku z danymi zwracany jest także kod błędu.

```
window.addEventListener("load",  function() {

  var apiStatus = document.getElementById('apiStatus');
  var apiKey = document.getElementById('apiKey').value;
  var city = document.getElementById('city').value;
  getJSON('http://api.openweathermap.org/data/2.5/weather?q='+ city +
'&units=metric&appid=' + apiKey,
  function(err, data) {
    if (err !== null) {
      document.getElementById("tableWeather").style.display = "none";
      if(err === 401)
      apiStatus.innerHTML = "Status API: <font color='red'> Zły klucz API</font>";
      else if(err === 404) apiStatus.innerHTML = "Status API: <font color='red'> Błędne
miasto</font>";
      else
       apiStatus.innerHTML = "Status API: <font color='red'> Nieznany błąd</font>";
    } else {
       apiStatus.innerHTML = "Status API: <font color='green'> Aktywny</font>";
        document.getElementById('temperatura').innerHTML = data['main']['temp']
        document.getElementById('cisnienie').innerHTML = data['main']['pressure']
        document.getElementById('wilgotnosc').innerHTML = data['main']['humidity']
        document.getElementById('pogoda').innerHTML = data['weather'][0]['icon']
        document.getElementById('pochmurnosc').innerHTML = data['clouds']['all']
```

```
        document.getElementById('wiatr').innerHTML = data['wind']['speed']
    }
  });
});
```

Druga funkcja uruchamiana jest po załadowaniu się strony. Z formularza pobrane są klucz API oraz miasto i używając funkcji getJSON odbierana jest odpowiedź serwera. Jeżeli serwer zwrócił błąd 401 oznacza to, że podano błędny klucz. Jeżeli 404 oznacza to że API nie mogło znaleźć podanego miasta. Jeżeli kod błędu był null, skrypt wyświetla komunikat o powodzeniu i wyświetla aktualne dane pogodowe do podglądu.

player.js

Plik zawiera skrypt umożliwiający odtworzenie streamu HLS w przeglądarce. Do jego stworzenia użyto biblioteki hls.js.

```
var video = document.getElementById('video');

function playM3u8(url){
          url = "http://192.168.56.101:2222/live/index.m3u8"
  if(Hls.isSupported()) {
     var hls = new Hls();
     var m3u8Url = decodeURIComponent(url)
     hls.loadSource(m3u8Url);
     hls.attachMedia(video);
     hls.on(Hls.Events.MANIFEST_PARSED,function() {
       video.play();
     });
   }
          else if (video.canPlayType('application/vnd.apple.mpegurl')) {
                    video.src = url;
                    video.addEventListener('canplay',function() {
                      video.play();
                    });
          }
}
```

Na początku pobierany jest handler do odtwarzacza HTML. Następnie w funkcji ustawiany jest adres streamu. W kolejnym kroku sprawdzane jest czy przeglądarka wspiera MediaSource Extensions. Jeżeli tak to tworzony jest nowy obiekt HLS, i stream przypisywany jest do odtwarzacza video HTML. Jeżeli nie to sprawdzane jest czy przeglądarka obsługuje vnd.apple.mpegURL. Jeżeli tak to do odtwarzacza HTML przypisywany jest adres streamu.

Tablica informacyjna

Kod tablicy informacyjnej został stworzony w języku Python. Składa się on z 3 elementów, wyświetlania mediów, wyświetlania paska z napisami oraz wyświetlanie zegara wraz z informacjami pogodowymi.

Scheduler.py

Plik udostępnia funkcje pobierające listę mediów oraz tekstów które mają zostać wyświetlone aktualnego dnia.

```
dbName = "tablica_informacyjna"
scheduleCol = "schedule"
scheduleTextCol = "textSchedule"

myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient[dbName]


# typy danych - image,video,hmtl,link,webcam
def scheduleMedia():
    dblist = myclient.list_database_names()
    if dbName not in dblist:
        return []

    mycol = mydb["schedule"]
    collist = mydb.list_collection_names()
    if scheduleCol not in collist:
        return []

    dateToday = datetime.today().strftime('%Y-%m-%d')
    query = {"$and": [{"start": {"$lte": dateToday}, "end": {"$gte": dateToday}}]}
    list = []
    for item in mycol.find(query, {"_id": 0}):
        list.append(item)
    return list
```

Pierwszą rzeczą jaką robimy jest ustalenie nazwy bazy danych oraz kolekcji. Następnie tworzymy nowe połączenie z bazą i wybieramy bazę danych. W funkcji scheduleMedia() sprawdzane jest, czy dana baza danych istnieje. Jeżeli nie zwracana jest pusta lista. Następnie patrzymy, czy istnieje dana kolekcja. Jeżeli nie, to także zwracana jest pusta

kolekcja. Jeżeli podane dane były prawidłowe pobieramy aktualną datę z systemu w formacie yyyy-MM-dd i tworzymy zapytanie wyświetlające elementy do wyświetlenia w dniu dzisiejszym. $lte oznacza “mniejszy lub równy” a $gte oznacza “większy lub równy”. Po wykonaniu się zapytania zwracana jest lista elementów do wyświetlenia.

weather.py

W pliku znajdują się funkcja pobierająca dane z OpenWeatherMap API o aktualnej pogodzie w danym mieście.

```
def get_current_weather(api_key,city):
    response = requests.get(
        'http://api.openweathermap.org/data/2.5/weather?q=' + city + '&units=metric&appid='
+ api_key)
    data = response.json()
    w_dict = {
        "temp": data['main']['temp'], # temperatura Celcius
        "pressure":  data['main']['pressure'],  # cisnienie hPa
        "humidity": data['main']['humidity'],  # wilgotnosc %
        "weatherIcon": data['weather'][0]['icon'],   # ikona pogody
        "clouds": data['clouds']['all'],    # pochmurnosc %
        "wind": data['wind']['speed']    # predkosc wiatru m/s
    }
    return(w_dict)
```

Jako argument funkcja przyjmuje klucz API oraz miasto. Po wywołaniu wysyła zapytanie do OpenWeatherApi z danymi podanymi w argumentach funkcji. Pobrane dane są w formacie json. Odpowiedź serwera następnie przetwarzana jest na słownik z danymi, które chcemy wyświetlić na pasku informacyjnym. Funkcja na końcu zwraca stworzony słownik.

Gallery.py

Skrypt zawarty w pliku odpowiedzialny jest za odtwarzanie mediów. Do wyświetlania elementów została użyta biblioteka OpenCV.

```
file = '/var/www/html/uploads/'


def video(filename):
    cap = cv2.VideoCapture(file+filename)

    while (cap.isOpened()):
        ret, frame = cap.read()
        print(ret)
        if ret:
            cv2.imshow('Gallery', frame)
            if cv2.waitKey(33) == 27:
                exit()
                break
        else:
            break
    cap.release()
```

Funkcja ta odpowiedzialna jest za wyświetlanie wideo. Na początku ustalany jest folder gdzie znajdują się przesłane pliki. Następnie funkcja jako argument przyjmuje nazwę pliku, którą przekazuje do funkcji VideoCapture. Pętla while będzie działać dopóki będą dostarczane nowe klatki wideo do wyświetlenia. Gdy otrzymano nową klatkę zmienna ret przyjmie wartość true. Gdy nowe klatki się skończą przyjmie ona wartość false. Wyświetlanie wideo można przerwać przedwcześnie naciskając klawisz ESC (kod 27).

```
cv2.namedWindow('Gallery', cv2.WND_PROP_FULLSCREEN)
cv2.setWindowProperty('Gallery', cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_FULLSCREEN)

l=scheduler.scheduleMedia()
print(l)
while True:
    for x in l:
        if x["type"] == 'image':
            img(x["name"], x["duration"])
        if x["type"] == 'video':
            video(x["name"])
        if x["type"] == 'url':
            u(x["name"], x["duration"])
        if x["type"] == 'webcam':
            camera(x["name"], x["duration"])
```

Na początku tworzone jest nowe okno o nazwie “Gallery” i ustawiane są jego parametry. Parametry ustawiają rozmiar okna odpowiadający rozdzielczości ekranu. Następnie z bazy danych pobierany jest aktualny harmonogram mediów. Po pobraniu elementów w nieskończonej pętli iterowane są kolejne elementy harmonogramu. Odpowiednie funkcje wywoływane są przez sprawdzanie wartości typu danego elementu.

infoBar.py

Skrypt odpowiedzialny jest za wyświetlanie paska z napisami pobranymi z bazy danych. Do stworzenia paska została użyta biblioteka Tkinter.

```
text_array = scheduler.scheduleText()
text_final = ''
for x in text_array:
    text_final += (x['text'] + ' * ')
```

Działanie skryptu rozpoczynamy od pobrania tekstów z bazy danych. Wszystkie napisy dodawane są do jednej zmiennej text_final. Teksty oddzielone są znakiem “*”.

```
root = tk.Tk()

root.wm_attributes('-type', 'splash')
deli = 90        # milliseconds of delay per character
svar = tk.StringVar()
labl = tk.Label(root, textvariable=svar, height=2,width=158,font=("Calibri",13))
root.geometry("+300+850")
```

Następnie tworzona jest instancja klasy Tk. W kolejnym kroku usuwamy górny pasek z nazwą programu i przyciskami minimalizacji, zmiany rozmiaru i zamknięcia. Potem ustalamy opóźnienie poruszania się liter na pasku informacyjnym. Wartość podawana jest w milisekundach. W kolejnej linijce tworzona jest zmienna biblioteki Tkinter i przypisana jest ona do etykiety. W etykiecie zmieniamy także wysokość, długość oraz czcionkę. Podpinamy ją także do głównego okna root. Root.geometry ustala położenie paska w formacie “+x+y”.

```
def shif():
    shif.msg = shif.msg[1:] + shif.msg[0]
    svar.set(shif.msg)
    root.after(deli, shif)

shif.msg = text_final
shif()
```

```
labl.pack()
root.attributes('-topmost', True)

root.mainloop()
```

Funkcja shif odpowiedzialna jest za poruszanie się tekstu na pasku. Przesuwa tekst w zmiennej o jeden znak, ustawia go do zmiennej svar a następnie ustawia opóźnienie na wartość zmiennej deli. shif.msg ustala jaki tekst będzie wyświetlany na pasku. Na koniec ustawiany jest atrybut -topmost co sprawia że pasek zawsze będzie widoczny i żadna aplikacja nie będzie mogła go zasłonić. Po konfiguracji uruchamiana jest nieskończona pętla.

infoBar.py

Plik zawiera kod zegara i wyświetlacza informacji o pogodzie. Tutaj także jest używana biblioteka Tkinter.

```
def tick():
    time_string = time.strftime("%H:%M:%S")
    clock.config(text=time_string)
    clock.after(200, tick)
```

Funkcja ta odpowiedzialna jest za odświeżanie zegara. W pierwszej linijce pobierany jest aktualny czas. W następnej zapisany czas wstawiany jest do odpowiedniej etykiety i na końcu ustawiane jest opóźnienie, z jakim kolejny raz uruchomi się funkcja.

```
def display_weather(current_info):
    if current_info < len(weather_info):
        weather_condition_label.config(text=weather_info[current_info][0])
        if weather_info[current_info][0] != "Pogoda":
            weather_condition_info.config(image='')
            weather_condition_info.config(text=weather_info[current_info][1])
        else:
            im = Image.open('icon/' + weather_info[current_info][1] + '.png')
            im = im.resize((20, 20), Image.ANTIALIAS)
            ph = ImageTk.PhotoImage(im)
            weather_condition_info.config(image=ph)
            weather_condition_info.image = ph

        weather_condition_label.after(2000, display_weather,current_info+1)
    else:
        current_info = 0
        weather_condition_label.config(text=weather_info[current_info][0])
        weather_condition_info.config(text=weather_info[current_info][1])
        weather_condition_label.after(2000, display_weather,current_info+1)
```

Funkcja ta odpowiedzialna jest za wyświetlanie aktualnych informacji o pogodzie. Na początku sprawdzane jest czy aktualny element nie wychodzi poza listę. Jeżeli tak to resetowany jest licznik i wyświetlany jest pierwszy element na liście. Jeżeli nie sprawdzane jest czy typ danej informacji pogodowej to “pogoda”. Jeżeli nie jest tego typu usuwana jest ikona (jeżeli była wyświetlona) i ustawiany jest nowy tekst opisujący jaki parametr pogody aktualnie jest wyświetlany i jego wartość. Jeżeli typ to “pogoda” zamiast wartości tekstowej wyświetlana jest ikona danej pogody. Po wczytaniu odpowiedniej ikony przypisywana jest ona do etykiety.

```
def update_weather():

    current_w = weather.get_current_weather(key,city)
    weather_info = weather.dict_to_string_tuple(current_w)
    weather_condition_info.after(1800000)
```

Funkcja ta odpowiedzialna jest za aktualizację danych pogodowych. Wywoływana jest ona co 30 minut.

Instrukcja użytkowania aplikacji

Autostart tablicy informacyjnej

Aby tablica informacyjna uruchamiała się automatycznie z uruchomieniem systemu należy utworzyć 3 pliki skryptowe .sh. Przechodzimy do folderu gdzie chcemy utworzyć nasze skrypty, tworzymy folder, w którym je umieścimy, przechodzimy do niego i uruchamiamy edytor tekstowy tworząc nowy skrypt.

```
cd /Home
mkdir Scripts
cd Scripts
nano gallery.sh
```

Po utworzeniu skryptu w nowo otwartym pliku wpisujemy:

gallery.sh

```
cd /Home/Desktop/Gallery
python3 gallery.py
```

I zapisujemy plik. Powtarzamy to dla dwóch kolejnych plików **infoBar.sh** i **infoBarWeather.sh**

infoBar.sh

```
cd /Home/Desktop/Gallery
python3 infoBar.py
```

infoBarWeather.sh

```
cd /Home/Desktop/Gallery
python3 infoBarWeather.py
```

Po utworzeniu plików naciskamy przycisk windows na klawiaturze i w nowo otwartym menu wpisujemy “Startup applications” i uruchamiamy aplikacje. Tutaj klikamy przycisk add i po kolei dodajemy skrypty naszej tablicy informacyjnej. W polu name nadajemy nazwę naszej aplikacji która ma się automatycznie uruchomić, w polu command podajemy ścieżkę do skryptu a w comment opcjonalny komentarz który opisuje skrypt. Po dodaniu wszystkich 3 skryptów tablica informacyjna uruchamiać się będzie automatycznie po restarcie systemu.

Włączenie nagrywania podglądu

Podgląd w panelu administratora nie jest automatycznie włączony. Można ten proces zautomatyzować w podobny sposób do uruchamiania samej tablicy informacyjnej. W utworzonym w poprzednim punkcie folderze tworzymy nowy skrypt i umieszczamy w nim kod

```
sudo /usr/local/nginx/sbin/nginx
ffmpeg -f x11grab -s 800x600 -r 30 -i $DISPLAY -vcodec libx264 -threads 0 -crf 0 -preset
ultrafast -f flv "rtmp://127.0.0.1:1935/live"
```

Pierwsza linijka uruchamia serwer nginx. Jeżeli użyty jest zewnętrzny serwer można tą linijkę pominąć. W drugiej linijce uruchamiane jest przechwytywanie obrazu z ekranu i przekazywanie go do serwera RTMP.

Po utworzeniu skryptu należy go dodać do autostartu. Przechodzimy do Startup applications i za pomocą przycisku add dodajemy skrypt do autostartu. Po ponownym uruchomieniu, obraz z Raspberry Pi powinien być widoczny w zakładce podgląd w panelu administratora.

Dodawanie mediów

Aby dodać nowe multimedia do wyświetlania na tablicy należy użyć panelu administratora. Aby dostać się do panelu administratora należy w przeglądarce wpisać adres IP Raspberry PI. Jeżeli nie znamy adresu IP należy wpisać polecenia ifconfig w terminalu Raspberry PI. Po zalogowaniu się w panelu administratora przechodzimy do zakładki Dodaj -> Media. Tutaj z rozwijanej listy musimy wybrać co chcemy dodać - obraz, film, adres do strony www, plik html czy adres do kamery IP. Następnie musimy w kolejnych polach określić od kiedy ma się wyświetlać dany zasób i kiedy jego wyświetlanie ma się zakończyć. Na końcu podajemy okres wyświetlania się wyrażony w sekundach. Jeżeli nie było żadnych konfliktów po naciśnięciu przycisku dodaj powinien wyświetlić się zielony komunikat o powodzeniu.

Dodawanie napisów do tablicy

Proces ten jest podobny do dodawania mediów tylko w tym przypadku w panelu administratora przechodzimy do zakładki Dodaj -> Teksty. Tutaj podajemy napis który ma się wyświetlić, oraz od kiedy ma się on wyświetlać i kiedy jego wyświetlanie ma się zakończyć. Po wpisaniu wszystkich danych akceptujemy je przyciskiem dodaj.

Usuwanie mediów i napisów

Jeżeli dodaliśmy jakiś zasób przypadkiem lub nie jest on już nam potrzebny, panel administratora udostępnia opcje usuwania zasobu. Aby usunąć zasób należy przejść do Listy odpowiednio “mediów” dla elementów multimedialnych i “tekstów” dla napisów. W ostatniej kolumnie każdego wiersza znajduje się znak X. Po naciśnięciu go zasób w danym wierszu zostaje usunięty z tablicy.

Zmiana miasta w wyświetlaczu pogody

Aby zmienić miasto z którego wyświetlana jest aktualna pogoda należy przejść do panelu administratora, a następnie do zakładki pogoda. Znajdują się tutaj dwa pola, Api Key oraz miasto. Aby zmienić miasto należy w polu miasto wpisać nowe miasto którego informacje pogodowe nas interesują. Po zaakceptowaniu zmian powinien wyświetlić się status czy informacje z danego miasta mogą być wyświetlone. Jeżeli tak to powinny zostać wyświetlone aktualne dane pogodowe które będą się także wyświetlać na tablicy.

Bibliografia

[1] Specyfikacja języka PHP v. 7.3.6, https://www.php.net/docs.php

[2] Opis biblioteki MongoDB do PHP, https://docs.mongodb.com/php-library/current/

[3] Opis sterownika MongoDB do PHP, https://docs.mongodb.com/ecosystem/drivers/php/

[4] Dokumentacja Composer v1.8.6, https://getcomposer.org/doc/

[5] Dokumentacja MongoDB v. 3.4.10, https://docs.mongodb.com

[6] Dokumentacja języka Python v. 3.7.3, https://docs.python.org/3/

[7] Dokumentacja OpenCV dla języka Python, https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

[8] Dokumentacja biblioteki Tkinter, https://docs.python.org/3/library/tk.html

[9] Dokumentacja Pymongo v. 3.8.0, https://api.mongodb.com/python/current/

[10] Opis biblioteki HTTP Requests wyd. 1.2.3, https://2.python-requests.org/en/master/

[11] Dokumentacja OpenWeatherMap API, https://openweathermap.org/api

[12] Dokumentacja serwera HTTP Live Stream NGINX v. 1.14.0, https://nginx.org/en/docs/

[13] Dokumentacja frameworku multimedialnego FFmpeg, https://ffmpeg.org/ffmpeg.html