

Dokumentacja



VI sem. Informatyka WE

Mateusz Norel (126901)

Tomasz Walczak (126851)

Mateusz Grabuszyński (126917)

Projekt zrealizowano w ramach przedmiotu Podstawy telekomunikacji.

Spis treści

Opis tematu	4
Uzasadnienie wyboru tematu projektu	4
Organizacja pracy	5
Podział prac między członków zespołu	5
Harmonogram prac	6
Opis funkcjonalności aplikacji	6
Funkcjonalności rozwiązania klienckiego	6
Funkcjonalności serwera	6
Opis endpointów	7
Stos technologiczny i wykorzystane narzędzia	8
Serwer	8
Swagger Editor	9
Flask	9
MongoDB	9
Klient (Raspberry Pi)	9
Chromium-browser	9
Scrot	9
Elementy stosu wspólne dla serwera i klienta	10
HTML oraz CSS	10
Responsive Web Design	10
Javascript, jQuery i wtyczki	10
Wytwarzanie oprogramowania	11
PyCharm, Nano, Vim i Notepad++	11
Git SCM	11
Wirtualne maszyny i serwery prywatne	11
Architektura rozwiązania	12
Baza danych	13
Schemat działania serwera	14
Schemat działania klienta	18
Instrukcja użytkowania aplikacji	20
Opis instalacji	20
Instalacja serwera	20
Pobranie, instalacja bazy MongoDB	20
Dodanie bazy systemu i kolekcji w bazie	21
Pobieranie zawartości repozytorium, instalacja bibliotek i uruchomienie serwera	21

Zatrzymywanie serwerów	21
Instalacja klienta	22
Opis użytkowania panelu administracyjnego	23
Strona główna	24
Zrzuty ekranu	25
Ustawienia sekcji głównej i prawej	25
Ustawienia paska news	26
Panel zarządzania plikami	26
Napotkane problemy	27
No 'Access-Control-Allow-Origin' is present (...)	27
Działo, ale przestało... (i nie wiadomo dlaczego)	27
Wnioski i przemyślenia z pracy nad systemem	28
Rozbudowa systemu	29
Zabezpieczenia	29
Odporność na braki sieci	29
Walidacja pól w managerze	29
Predefiniowane elementy rotatora	29
Składanie propozycji poprawek	30
Przypisy końcowe	31

Opis tematu

Tematem projektu jest utworzenie oprogramowania systemu zarządzającego siecią składającą się z tablic informacyjnych przy użyciu serwera sterującego.

Tablica informacyjna oparta na komputerze jednopłytkowym Raspberry Pi służy do wyświetlania w odpowiednich rotatorach umieszczonych na serwerze informacji, odtwarzania plików wideo, wyświetlania obrazów i stron internetowych. Rotatorem nazywamy fragment strony cyklicznie wyświetlający różne informacje z podanego zakresu przez określony dla nich czas. Nasze rozwiązanie składa się z trzech takich elementów. Lewego (dalej zwanego także głównym), prawego – służącego do wyświetlania informacji pobocznych oraz panelu informacji (newsbara).

Głównym zadaniem serwera jest przechowywanie informacji wyświetlanych przez tablice informacyjną. Ponadto, umieszczony na nim panel administracyjny (dalej także manager), pozwala na edycję treści rotatorów a także zmianę zasad ich wyświetlania.

Uzasadnienie wyboru tematu projektu

Jako zespół, zainteresowaliśmy się tematem tablic informacyjnych ze względu na rosnące zainteresowanie i zapotrzebowanie użytkowników końcowych na systemy podobnego typu. Przykładem mogą być systemy informacji uczelnianej (w tym składający się z blisko 50 monitorów System Informacji Wewnętrznej Politechniki Poznańskiej^[1]) czy pracowniczej, a także bardziej spersonalizowane systemy typu Amazon Echo Show^[2], Echo Spot^[3] czy Google Smart Display zapowiadany na konferencji Google/IO^[4] oraz inteligentne lustra^[5], które coraz częściej pojawiają się w mieszkaniach i hotelach.

Wcześniejsze pozytywne doświadczenia oraz fakt posiadania komputerów jednopłytkowych Raspberry Pi przez dwóch z trzech członków zespołu, skłonił nas do przyjrzenia się oferowanym rozwiązaniom komercyjnym, w których są one używane. Miejski Zakład Komunikacji we Wrocławiu używa komputera Raspberry Pi w nowych kasownikach biletów^[6]. Nie jest to jednak jedyny przykład, kanadyjska firma Bryston zbudowała na bazie komputera wysokiej jakości odtwarzacz muzyki o nazwie BDP-Pi Digital Player^[7], a firma AntMiner koparkę kryptowalut^[8].

Jako wyzwanie postawiliśmy sobie scentralizowane zarządzanie aplikacjami uruchomionymi na wielu urządzeniach. Ostatecznie zrezygnowaliśmy z pisania aplikacji na urządzenia końcowe, na rzecz uruchamiania przeglądarki internetowej zintegrowanej ze scentralizowanym RESTowym API. Dzięki takiemu rozwiązaniu,

naszą implementację można uruchomić zarówno na Raspberry Pi, jak i na komputerze domowym czy innym rozwiązaniu jednopłytkowym.

Organizacja pracy

Podział prac między członków zespołu

Pracę staraliśmy się rozdzielać między sobą w sposób sprawiedliwy. Niestety nie zawsze było to możliwe, głównie ze względu na różnice w znajomości stosowanych rozwiązań oraz wysoki stopień zaangażowania w inne projekty prowadzone przez członków zespołu. Duża część zadań była, w końcowym momencie realizacji projektu, prowadzona przez każdego z nas.

W tabeli podano pierwsze litery imion i nazwisk członków zespołu, zamiast ich pełnych odpowiedników.

Przydzielone zadania	Członkowie zespołu
Zapoznanie z możliwościami Raspberry Pi w celu doboru odpowiedniego stosu technologicznego	MN, TW, MG
Utworzenie repozytorium Github	MG
Zakup serwera VPS celem ułatwienia prac deweloperskich	MG
Budowa API w języku YAML (editor.swagger.io)	MN, TW, MG
Kod backendu RESTowego API (Python)	MN, MG
Kod backendu managera (Python)	MN, MG
Kod frontendu managera (HTML, CSS, JS)	MG, MN, TW
Kod backendu managera (JS, jQuery)	TW, MG, MN
Opracowanie skryptów uruchamiających usługi i programy na Raspberry Pi	MN, MG
Kod frontendu klienta (HTML, CSS)	MG, TW
Kod backendu klienta (JS)	MN, MG, TW
Tworzenie prezentacji roboczych projektu	MN, TW, MG
Dokumentacja projektu	MN, MG, TW

Tabela 1: Podział zadań.

Harmonogram prac

Ustalenie harmonogramu prac polegało na przypisaniu zadań do odpowiednich terminów zajęć, odbywających się w interwale dwutygodniowym (z większymi przerwami podczas świąt). Niestety, co było widać już zaledwie po trzech prezentacjach, nie trzymaliśmy się tego harmonogramu. Zadziałała zasada zbliżającego się deadline'u, przez co większość projektu powstała tak naprawdę w ostatnich dwóch tygodniach. Nie jest to dobre podejście programistyczne – powoduje powstawanie dużej liczby błędów w kodzie.

Opis funkcjonalności aplikacji

W poniższych podrozdziałach znajduje się lista funkcjonalności rozwiązania klienckiego i serwerowego, wraz z ich krótkim opisem.

Funkcjonalności rozwiązania klienckiego

Zadaniem rozwiązania stworzonego dla Raspberry Pi jest wyświetlanie trzech rodzajów rotatorów:

- Rotator główny (lewy) oraz prawy pozwalają na wyświetlenie strony WWW oraz obrazu z adresu URL, filmu lub obrazu na żywo hostowanego w serwisie YouTube, a także grafiki i plików HTML trzymany na serwerze;
- Rotator wiadomości (newsbar) wyświetla obecną datę i godzinę, a także informacje tekstowe znajdujące się w bazie danych serwera. Możliwa jest zmiana koloru tła tego elementu na ciemnoczerwony w przypadku ważnych wiadomości.

Klient okresowo przesyła serwerowi zrzuty ze swojego ekranu, celem ewentualnej scentralizowanej diagnostyki.

Funkcjonalności serwera

Przygotowana aplikacja managera pozwala na:

- Wyświetlenie strony głównej managera wraz z menu z odnośnikami do wszystkich podstron managera, które pozwalają na wygodne zarządzanie treścią prezentowaną w tablicy informacyjnej;
- Pokazanie podstrony zawierającej zrzuty ekranu pochodzące ze wszystkich klientów, którzy identyfikowani są poprzez nazwę znajdującą się pod każdym zrzutem ekranu;
- Pobranie podstrony umożliwiającej zarządzanie lewym rotatorem, a na niej:
 - Zmianę kolejności wyświetlania elementów;
 - Zmianę treści wyświetlanych elementów;

- Zmianę okresu wyświetlania elementów;
- Zmianę długości pojedynczego wyświetlenia elementu;
- Usuwanie elementów z listy wyświetlania.
- Odwiedzenie podstrony zarządzania lewym rotatorem, która posiada te same funkcjonalności co odpowiadająca strona dla administracji lewego rotatora;
- Wyświetlenie podstrony zawierającej managera belki informacyjnej, a na niej:
 - Zmianę kolejności wyświetlania elementów;
 - Zmianę głównej treści (większy napis);
 - Zmianę treści pobocznej (mniejszy napis);
 - Zmianę okresu wyświetlania;
 - Zmianę długości pojedynczego wyświetlenia;
 - Ustawienie ważności elementu;
 - Usuwanie elementów z listy.
- Pokazanie podstrony zarządzania plikami, a na niej:
 - Dodawanie plików;
 - Przeglądanie dodanych plików wraz z ich nazwą pierwotną i nazwą pliku umieszczonego na serwerze;
 - Usuwanie plików.

Wszystkie podstrony zawierają także odnośniki do strony głównej (poza samą stroną główną) oraz do repozytorium i informacji o autorach projektu.

Wykorzystywaną bezpośrednio przez klienta funkcjonalnością serwera jest pobieranie z endpointa plików JSON wraz ze wszystkimi elementami rotatorów.

Opis endpointów

Dokładna rozpiska wszystkich endpointów oferowanych przez serwer znajduje się w tabeli poniżej.

endpoint	metoda	krótki opis
/api/display	GET	pobranie pliku JSON z elementami do rotatorów
/api/display/{display_id}	GET	nieużywana (pierwotnie miała służyć do pobierania screenów z urządzeń o danym display_id)
/api/display/{display_id}	PUT	wykorzystywany przez klientów do umieszczania najnowszych zrzutów ekranu (na zasadzie podmiany)
/api/files/{hash}	GET	pobieranie pliku o podanym hashu

/display	GET	przekierowanie do strony wyświetlacza (display.html)
/manager	GET	przekierowanie do strony głównej managera (manager.html)
/manager/displays	GET	przekierowanie do podstrony z dostępnymi wyświetlaczami (display.html)
/manager/files	GET	przekierowanie do podstrony z panelem zarządzania plikami (files.html)
/manager/files	POST	dodawanie pliku do bazy danych i katalogu na serwerze
/manager/files/{hash}	DELETE	usuwanie pliku o podanym hashu
/manager/main_content	GET	wyświetlanie podstrony dotyczącej zarządzania lewym rotatorem
/manager/main_content	POST	podmiana elementów lewego rotatora
/manager/news_bar	GET	wyświetlenie podstrony zarządzania belką informacyjną
/manager/news_bar	POST	podmiana elementów wyświetlanych w belce informacyjnej
/manager/right_panel	GET	wyświetlanie podstrony dotyczącej zarządzania prawym rotatorem
/manager/right_panel	POST	podmiana elementów prawego rotatora

Tabela 2: Opis wszystkich endpointów oferowanych przez serwer.

UWAGA: Należy pamiętać, że aby dostać się do bezpośrednio konkretnego endpointa należy podać także odnośnik do wersji (/v1) na początku ścieżki.

Stos technologiczny i wykorzystane narzędzia

Serwer

Rozwiązanie serwerowe zbudowano z elementów opisanych w poniższych podrozdziałach. Każda technologia została krótko opisana, a jej wybór poparty argumentami.

Swagger Editor

Dostępne pod adresem editor.swagger.io narzędzie, służące do prostego pisania API RESTowych przy użyciu języka YAML i generowania serwera oraz klienta takiego serwera w wielu popularnych językach programowania, w tym w Pythonie.

Zostało ono wybrane ze względu na jego przejrzystość, prostotę języka YAML i dynamiczną walidację błędów, co pozwoliło znacznie skrócić budowanie komponentów serwera. Dodatkowo jeden z członków zespołu posiadał wcześniejsze pozytywne doświadczenia związane z używaniem tego narzędzia.

Flask

Microframework dla języka Python wydany na licencji BSD^[9], bazujący na Werkzeug i Jinja 2, służący do tworzenia i uruchamiania serwera RESTowego.

Wybraliśmy Flaska ze względu na prostą instalację i uruchomienie, a także obszerną i przejrzystą dokumentację.

MongoDB

Zdecydowano się na dokumentową bazę danych MongoDB. Wybór ten był podyktowany wysyłaniem danych do ekranów w postaci dokumentów JSON, które mogą być bezpośrednio przechowywane w bazie, bez potrzeby dodatkowego przetwarzania czy tłumaczenia.

Klient (Raspberry Pi)

Rozwiązanie klienckie zostało przygotowane tak, aby instalacja bez gotowego obrazu całego systemu wymagała jak najmniejszej liczby kroków wdrażającego użytkownika.

Chromium-browser

Przeglądarka internetowa o otwartym kodzie źródłowym, która zgodnie z zapisami na stronie projektu^[10], skonstruowana została aby szybciej, bezpieczniej i stabilniej doświadczać surfowania w sieci. Została wybrana ze względu na prostotę instalacji wymagającą wykonania jednego polecenia (nie licząc aktualizacji aptitude) i możliwość odpalenia w trybie kiosku.

Scrot

Proste narzędzie służące do przechwytywania i zapisywania zrzutów ekranu urządzenia^[11]. Instalowane jednym poleceniem, pozwala na zapis obrazów w różnym formacie, nadając im podaną nazwę oraz tworząc dodatkową miniaturkę

(tzw. *thumbnail*). Istotnym aspektem jest także przełącznik `-d`, pozwalający na ustawienie opóźnienia wykonywania zrzutu oraz `-q`, umożliwiający ustawienie jakości wykonywanego zrzutu.

Elementy stosu wspólne dla serwera i klienta

HTML oraz CSS

HTML (Hypertext Markup Language) jest językiem wykorzystywanym do projektowania szablonów stron WWW, a CSS (Cascading Style Sheet) do ich stylizowania.

Zdecydowano się na użycie ich kombinacji z racji wykorzystania przeglądarki jako silnika wyświetlania elementów na tablicach. Nie wykorzystywano preprocesorów takich jak SASS czy LESS^[12] gdyż nie były one konieczne w tak prostym zastosowaniu.

Responsive Web Design

Pierwsze wersje szablonu tablicy informacyjnej nie “składały się” odpowiednio do rozdzielczości na, której były wyświetlane o czym przekonaliśmy się podczas prezentacji systemu na jednej z prezentacji przeprowadzonych podczas zajęć. Z tego powodu zastosowaliśmy Responsive Web Design, dzięki któremu, napisy z newsami, zegarem i datą są wyświetlane poprawnie niezależnie od rozdzielczości.

```
108 @media only screen and (max-width: 1350px){
109     #time{
110         font-size: 2em
111     }
112     #date{
113         font-size: 0.75em
114     }
115     #info-title{
116         font-size: 2em
117     }
118 }
119 }
```

Listing 1: Fragment kodu CSS reprezentującego dopasowanie elementów do ekranów o maksymalnej szerokości 1350 px.

Elementy managera również są częściowo napisane tak, aby były responsywne.

Javascript, jQuery i wtyczki

Język i biblioteka szeroko stosowane w technologiach webowych. U nas zostały zastosowane w backendzie i frontendzie managera oraz wyświetlaczy.

Tryb przeciągnij i upuść elementów managera obsługuje wtyczka do jQuery o nazwie *TableDnD* w wersji 0.8^[13]. Rozwijany kalendarz do wyboru okresu to zaś skrypt JS o nazwie *Date Range Picker*^[14].

Rotator strony opiera się na skrypcie pobierającym z API dane przy użyciu polecenia `jQuery.getJSON()`. Wykorzystano także skrypt *simpleClock* v1.0^[15] udostępniony przez firmę *TICKTOO* na licencji MIT. Jest to bardzo prosty zegarek napisany w JavaScriptcie, który widać w lewym dolnym rogu wyświetlacza, w newsbarze. Przed użyciem należało zmienić format daty oraz spolszczyć dni tygodnia i miesiące, a także ustalić format wyświetlania.

Wytwarzanie oprogramowania

W podrozdziale przedstawione zostaną programy użyte do programowania i zarządzania plikami.

PyCharm, Nano, Vim i Notepad++

Kod programu napisany w języku Python, powstawał w środowisku programistycznym *JetBrains PyCharm* w wersji 2018.1.1 Community Edition^[16]. Pozostałe pliki powstawały przy użyciu różnych edytorów tekstu, dobranych w zależności od upodobań członka zespołu pracującego aktualnie nad danym elementem.

Git SCM

Do obsługi repozytoriów wykorzystano darmowy program *Git SCM* dostępny na stronie projektu^[17]. Mechanizm rozgałęziania był wykorzystywany tylko częściowo. Większość commitów była wrzucana bezpośrednio na mastera.

Wirtualne maszyny i serwery prywatne

Podjęto się zakupu wirtualnego serwera prywatnego (ang. *Virtual Private Server*), celem prostszej pracy przy rozwoju aplikacji. Łączono się do niego poprzez protokół SSH programem *PuTTY*^[18]. To rozwiązanie sprawdzało się tylko częściowo, gdyż nie oferowało wszystkich wygod pracy na systemie lokalnym. Błędna konfiguracja spowodowała także problemy przy połączeniu z systemem wersjonowania *GitHub*.

Z racji, iż jeden z członków zespołu jako główny system operacyjny używa na swoich komputerach systemu *Windows*, potrzebne było zainstalowanie odpowiednich maszyn wirtualnych z odpowiednio dobraną dystrybucją systemu *Linuxowego*. Jako rozwiązanie wirtualizujące wybrano program *Oracle VM VirtualBox*^[19]. Na maszynach zainstalowano system *Ubuntu* w wersji 17.10^[20],

gdyż jest to najbardziej zbliżony system do tego, którego używa zakupiony serwer prywatny. Pozostali członkowie zespołu pracowali na lokalnych maszynach.

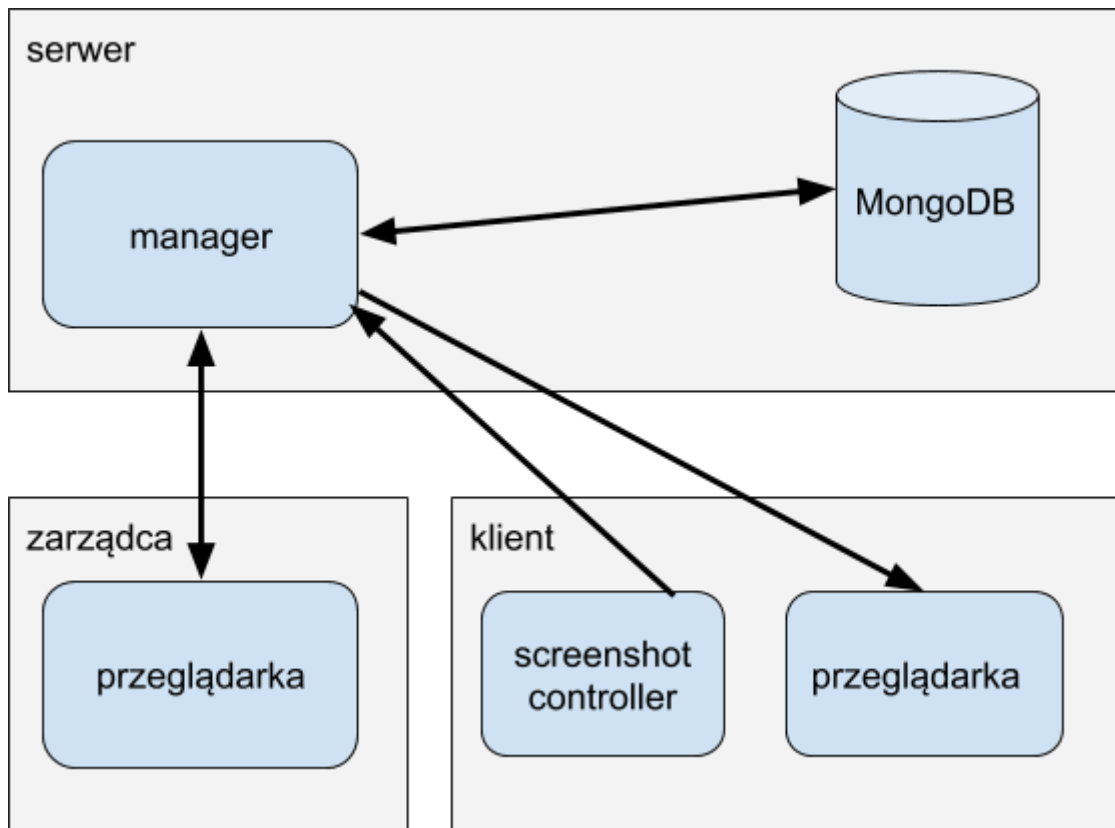
Architektura rozwiązania

System bazuje na architekturze typu klient-serwer. Zarządca pełni rolę poboczną, korzystając z managera do wprowadzania ustawień systemu i podglądu zrzutów ekranów serwowanych przez klientów.

W roli serwera występuje system Ubuntu 17.10 w wersji serwerowej. Jego oprogramowanie składa się z konsolowego programu renderującego i obsługującego API oraz managera, domyślnie dostępnych na porcie 8080, a także bazy danych MongoDB w postaci katalogów oraz plików. Manager służy do przekazywania informacji pomiędzy serwerem a klientem. Do bazy danych MongoDB można teoretycznie dostać się bezpośrednio z zewnątrz, lecz rozwiązanie to nie jest użyte w naszym systemie.

Klientami są urządzenia RaspberryPi z zainstalowanym systemem Raspbian, które łączą się do serwera, a następnie wyświetlają stronę w przeglądarce w trybie pełnoekranowym. Klient pobiera informacje z serwera i wyświetla je w graficznym interfejsie użytkownika (przeglądarce internetowej) w odpowiedniej, obrobionej wcześniej, formie. Podaje on także serwerowi zrzuty swojego ekranu, które później można wyświetlić w odpowiedniej zakładce managera. Poniżej przedstawiono uproszczony schemat rozwiązania.

W pewnym momencie pojawił się pomysł zaimplementowania rozwiązania bazującego na WebSocketach^[21], które wymieniałyby dane z serwerem zdecydowanie częściej, z praktycznie pomijalnym opóźnieniem, jednakże odrzuciliśmy go ze względu na konieczność wprowadzenia zbyt dużych zmian po stronie zarówno klienta jak i serwera.



Rysunek 1: Architektura infostradki. Strzałki wskazują kierunek przepływu informacji.

Baza danych

MongoDB jako rozwiązanie bazodanowe pozwala przechowywać dokumenty JSON o nieokreślonej strukturze. Baza danych podzielona jest na kolekcje, w których przetrzymywane są obiekty składające się na prezentowany komunikat lub wspomagające pracę systemu rozszerzając go o dodatkowe funkcjonalności. Poniżej znajduje się krótki opis wszystkich kolekcji. W opisie pominięto standardowe identyfikatory nadawane każdemu dokumentowi w bazie.

- **left** - przechowuje obiekty, które będą prezentowane w głównym, największym oknie;

- + content
 - + source - źródło do pliku bądź adres strony WWW
 - + subtitles - tymczasowo nieużywane, pierwotnie miało służyć dodawaniu napisów do filmów
- + duration - długość pojedynczego wyświetlenia (w sekundach)
- + since - od którego momentu rozpocząć wyświetlanie elementu (znacznik czasu w formacie YYYY-MM-DD hh:mm:ss)
- + until - do kiedy go wyświetlać (format jak wyżej)
- + type - typ elementu (np. "www", "yt", "file")

- **right** - przechowuje obiekty, które będą wyświetlane w wąskim oknie po prawej stronie. Jej struktura jest taka sama jak kolekcji **left** opisanej powyżej;
- **news** - przechowuje wiadomości tekstowe do wyświetlenia w dolnym pasku;

- + **title** - większy napis na belce wiadomości
- + **content** - mniejszy napis na belce wiadomości
- + **important** - marker ważności wiadomości. Ustawienie wartości na **true** powoduje zmianę koloru belki wiadomości na czerwony.
- + **duration** - czas pojedynczego wyświetlenia (w sekundach)
- + **since** - znacznik czasu
- + **until** - znacznik czasu

- **scrshots** - przechowuje zrzuty ekranu z podłączonych urządzeń;

- + **display_name** - nazwa wyświetlacza, z którego pochodzi zrzut
- + **display_id** - id tegoż urządzenia
- + **screenshot** - grafika zakodowana w base64

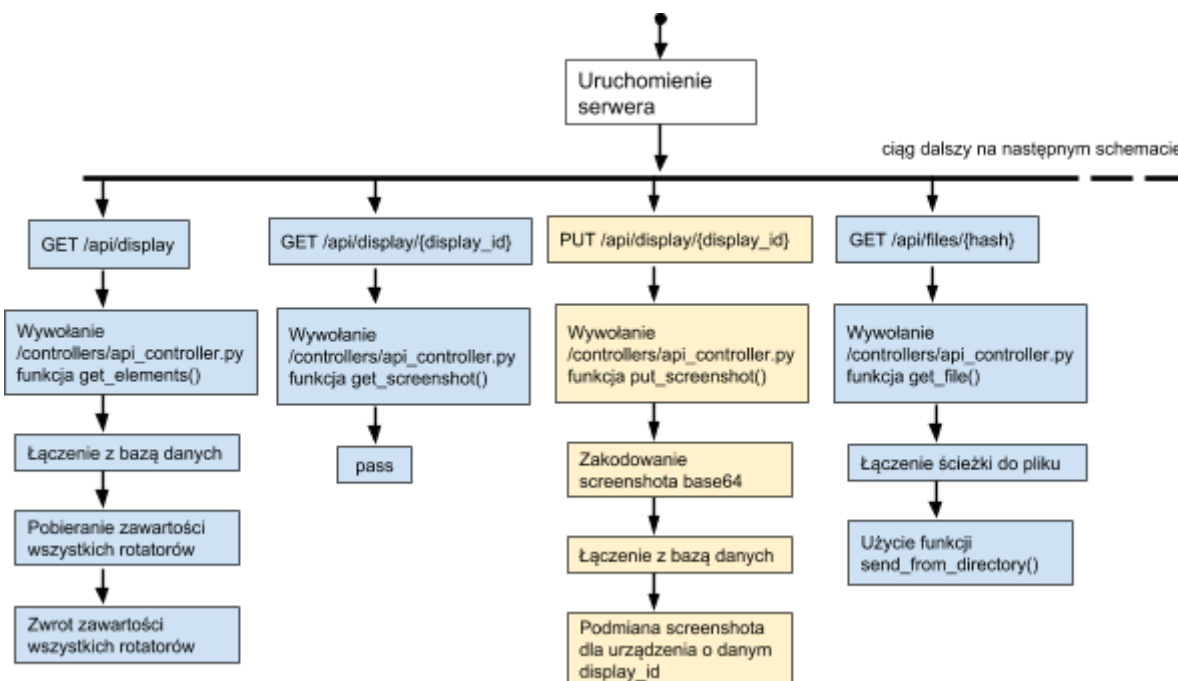
- **files** - przechowuje pliki wysłane przez użytkowników, można je wykorzystać do wyświetlenia w rotatorach.

- + **type** - typ elementu przechowywanego na serwerze
- + **name** - nazwa pliku, który nadesłał użytkownik menedżera
- + **hash** - nazwa pliku faktycznie przechowywanego na serwerze

Jedyną, zależną od ustawionego typu, relacją pomiędzy kolekcjami jest połączenie poprzez **hash** serwowanego pliku poddrzewa **content** z kolekcji **right** lub **left** oraz **hash** z kolekcji **files**.

Schemat działania serwera

Z racji mnogości funkcji, zdecydowano się na pociętą wersję uproszczonego schematu działania. Kolejne obrazki należy traktować jakby były ułożone na schemacie obok siebie. Zatrzymanie serwera powoduje brak odpowiedzi na zapytanie kierowane na którykolwiek z przedstawionych endpointów.



Rysunek 2: Pierwsza część schematu działania serwera.

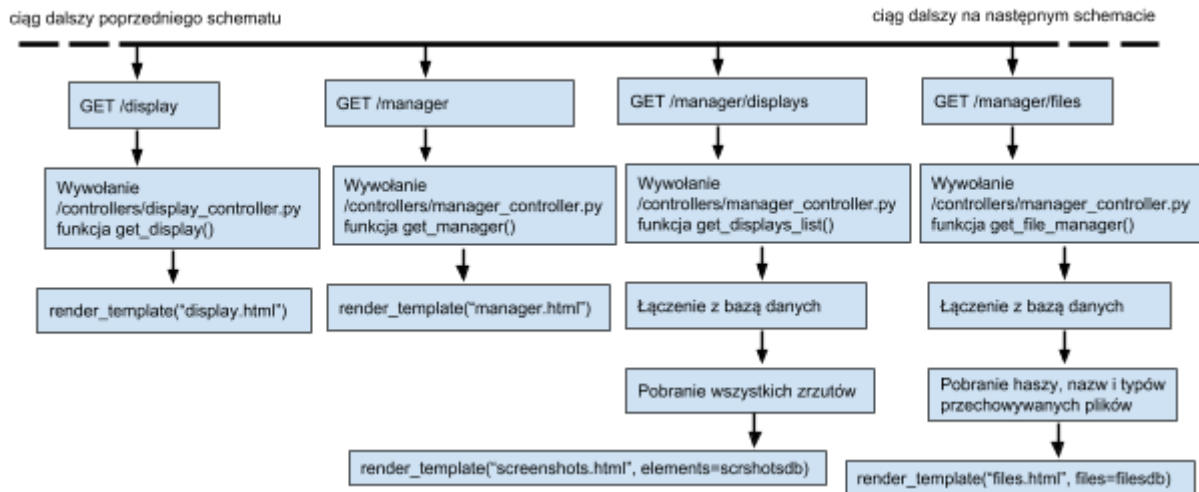
W pliku `/controllers/api_controller.py` znajdują się jedynie 4 definicje funkcji. Jedną z nich, `get_screenshot()`, to czysty `pass`, czyli przejście dalej. Jest tak, gdyż implementacja tej funkcji okazała się być niepotrzebna, co okazało się dopiero w późnych etapach implementacji. Pozostałe funkcje, wykonują swoje akcje poprawnie. Ciekawym listingiem może być fragment funkcji `put_screenshot()`, zwłaszcza linia 3 odpowiadająca za zakodowanie screenshota do `base64`. Z racji, iż obraz przychodzi do serwera w postaci pliku, zostaje odczytany `screenshot.read()`, zakodowany `base64` `base64.b64encode(...)` i celem umieszczenia w bazie danych zakodowany do ASCII poprzez `.decode('ascii')`.

```

47 def put_screenshot(display_id, display_name, screenshot):
48     #screenshot to base64
49     ssencoded = base64.b64encode(screenshot.read()).decode('ascii')
50
51     #db handler
52     client = MongoClient(MONGO_HOST, MONGO_PORT)
53     db = client.infostradka #select db infostradka
54
55     #db upload into scrshots collection
56     db.scrshots.update({"display_id": display_id,
57                        {"display_id": display_id, "display_name": display_name,
58                         "screenshot": ssencoded}, upsert=True)
57     pass

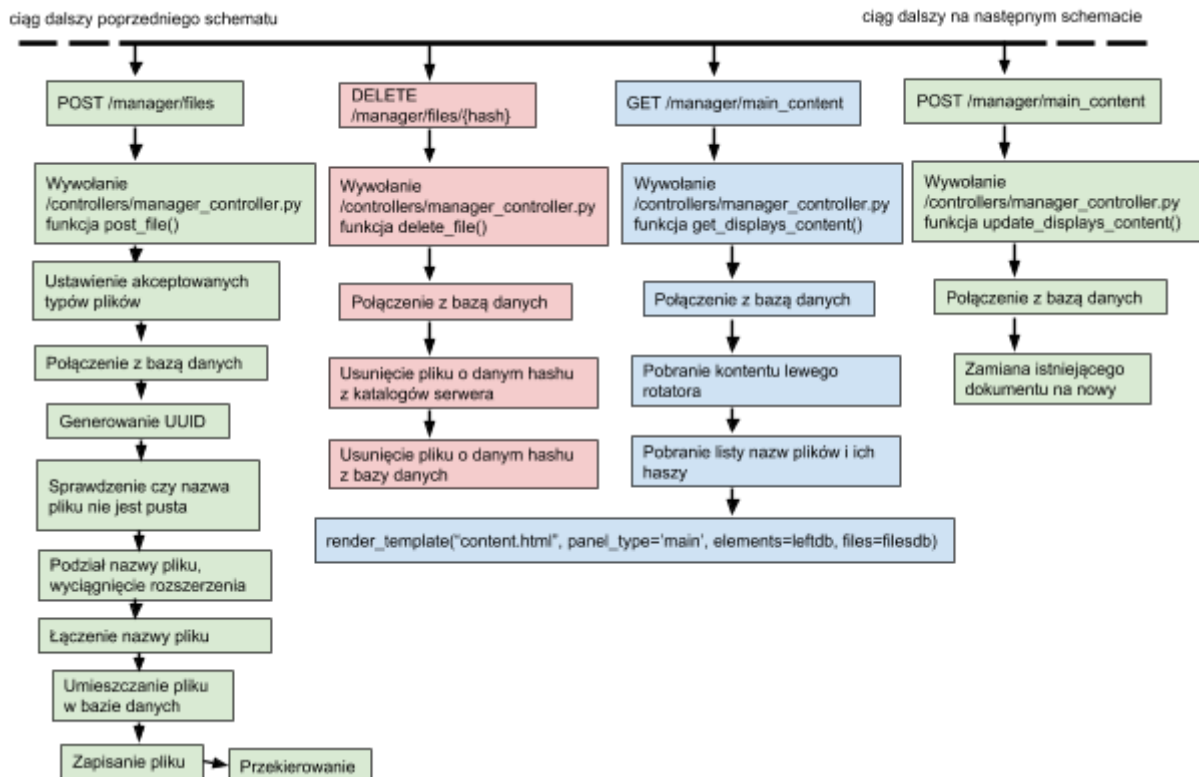
```

Listing 2: Funkcja `put_screenshot()` w pliku `/controllers/api_controller.py` z widocznym kodowaniem plików obrazów do `base64`.



Rysunek 3: Druga część schematu działania serwera.

W pliku `/controllers/display_controller.py` znajduje się jedynie funkcja renderująca stronę dla klienta (wyświetlacza). Plik `/controllers/manager_controller.py` jest najdłuższym plikiem skryptu uruchamianym na serwerze. Funkcje `get_displays_list()` oraz `get_file_manager()` pobierają z bazy danych odpowiednie informacje, odpowiednio zrzuty ekranu i listę plików, by później zwrócić je jako parametry w adresie strony.

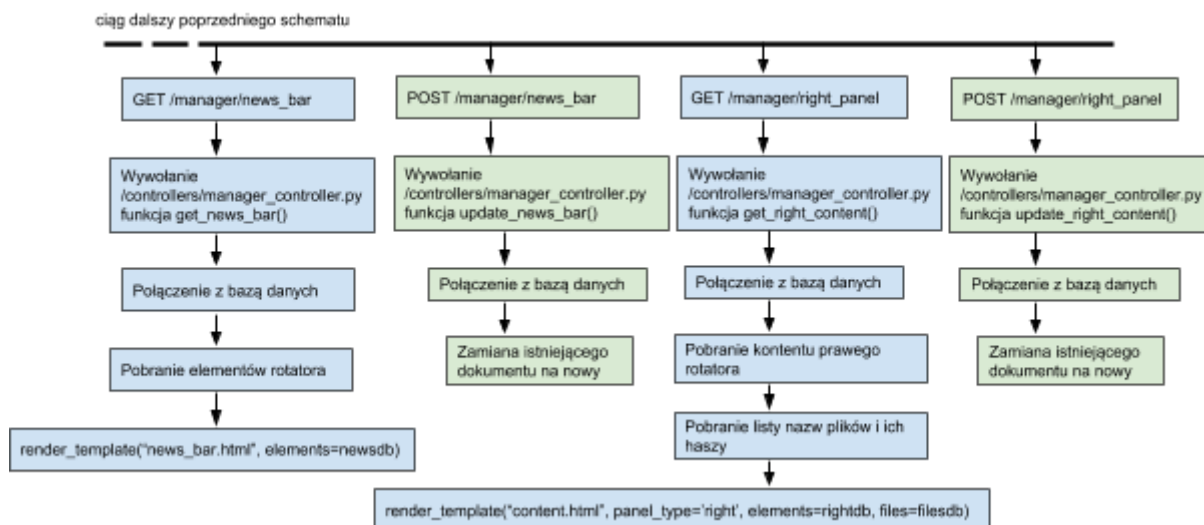


Rysunek 4: Trzecia część schematu działania serwera

Funkcja `post_file()` jest najbardziej rozbudowaną funkcją managera, stąd zostanie częściowo wylistowana i omówiona poniżej. Ustawia ona zbiór (typ `set`) akceptowanych typów plików `ALLOWED_EXTENSIONS`, łączy się z bazą danych, tworzy nowy uuid `hash=uuid.uuid4()`. Następnie sprawdza czy nazwa pliku nie jest pusta, jeśli jest to zwraca błąd `?err=emptyfilename`, jeśli nie sprawdza czy typ pliku jest prawidłowy. Jeśli nie jest, zwraca błąd `?err=forbiddenfiletype`. Gdy zaś jest, sprowadza rozszerzenie pliku do zmiennej `fctype = file.filename.split('.')[1].lower()` i podmienia niepoprawne typy MIME na poprawne. Następnie odbywa się łączenie nazwy pliku jako `hash`, kropka i typ pliku `partpath = str(hash)+'.'+fctype`. Na końcu znajduje się insert do kolekcji `files` w bazie danych, łączenie całkowitej ścieżki dostępu do zapisu pliku w katalogach serwera i zapis tego pliku pod tą ścieżką. Poprawne przejście przez sekwencję prowadzi do redirecta do tego samego miejsca bez żadnego komunikatu.

```
127 def post_file(file):
128     ALLOWED_EXTENSIONS = set(['html', 'htm', 'png', 'jpg', 'jpeg',
                                'gif'])
129     (...)
130     #hash = uuid
131     hash = uuid.uuid4()
132     print(hash)
133
134     if file.filename == '':
135         return redirect("/v1/manager/files?err=emptyfilename")
136     if file and allowed_file(file.filename, ALLOWED_EXTENSIONS):
137         fctype = file.filename.split('.')[1].lower()
138         if fctype == 'htm':
139             fctype = 'html'
140         elif fctype == 'jpg':
141             fctype = 'jpeg'
142
143         partpath = str(hash)+'.'+fctype
144
145         db.files.insert({"type": fctype,
146                         "name": file.filename, "hash": partpath})
147         fullpath = os.path.join(os.getcwd(), FILES_DIR, partpath)
148         file.save(fullpath)
149         print('file saved as:' + fullpath)
150
151         return redirect("/v1/manager/files")
152     else:
153         return redirect("/v1/manager/files?err=forbiddenfiletype")
```

Listing 3: Fragment funkcji `post_file()` w pliku `/controllers/manager_controller.py`.



Rysunek 5: Ostatnia, czwarta część schematu działania serwera.

Powyższe funkcje znajdujące się w `/controllers/manager_controller.py` są prawie takie same jak funkcje dla innych rotatorów. Posty odbywają się przy użyciu jednej funkcji wspólnej `db_replace()` przyjmującej jako pierwszy parametr numer rotatora, którego zawartość ma podmienić.

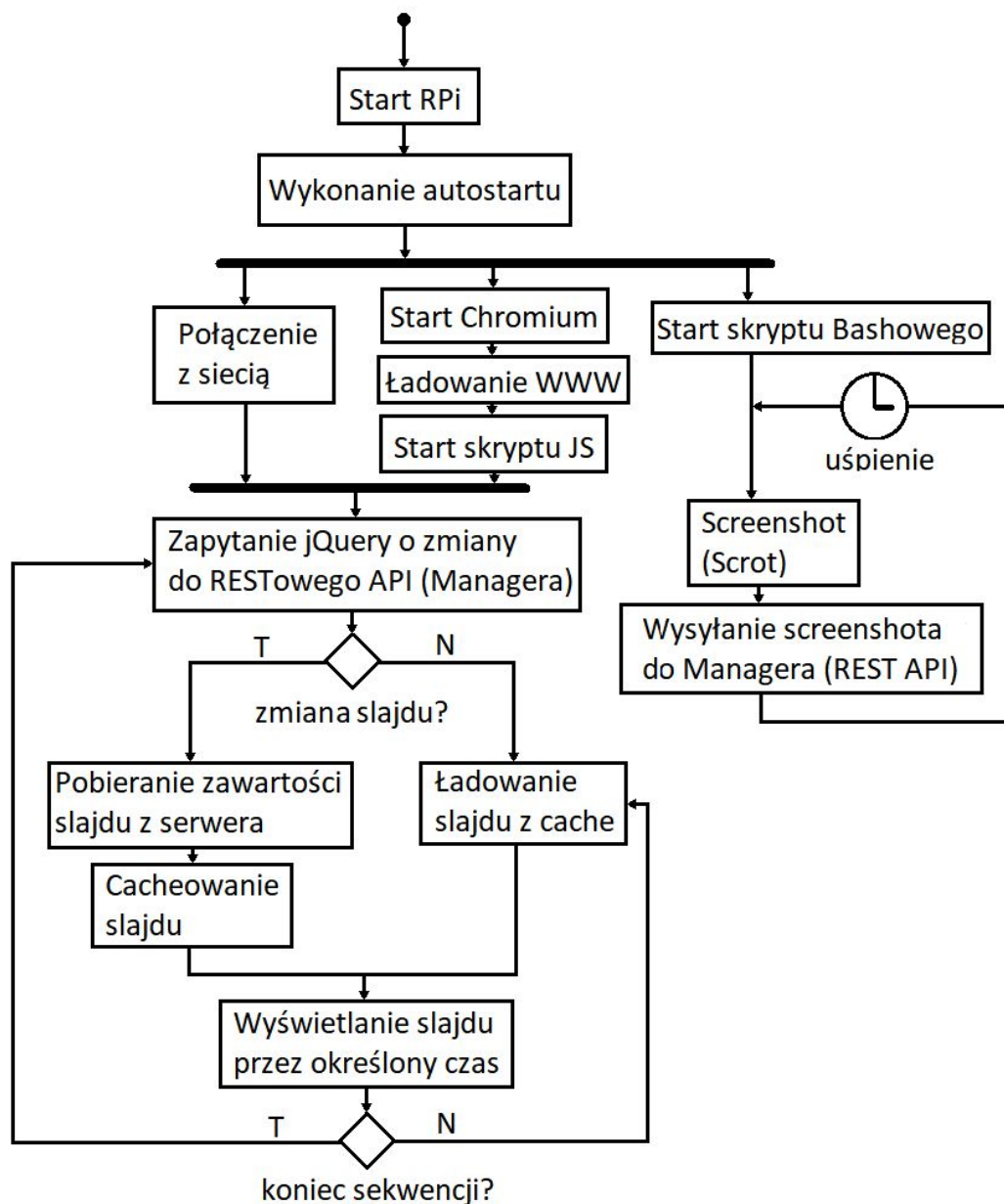
```

15 def db_replace(rotator, body):
16     client = MongoClient(MONGO_HOST, MONGO_PORT)
17     db = client.infostradka
18     if rotator == 1:
19         db.left.remove()
20         db.left.insert(body)
21     if rotator == 2:
22         db.right.remove()
23         db.right.insert(body)
24     if rotator == 3:
25         db.news.remove()
26         db.news.insert(body)
  
```

Listing 4: Funkcja `db_replace()` w pliku `/controllers/manager_controller.py` z widocznym podmianianiem bazy danych (sekwencja `remove-insert`).

Schemat działania klienta

Poniższy schemat przedstawia w jaki sposób działa klient. Serwer jest odpytywany przez jQuery o dokumenty co 5 minut. Screenshoty są tworzone i przesyłane co 5 sekund. Slajdy (elementy rotatora) są cache'owane przez przeglądarkę, jednakże w przypadku utraty dostępu do sieci, rotator zawiesza się i nie pokazuje następnych slajdów (co nie było zamierzone).



Rysunek 6: Schemat działania klienta.

Zapisany poniżej skrypt przedstawia początek (dotyczący lewego rotatora) głównej funkcji rotatorów, która sprawdza czy plik JSON został poprawnie pobrany z serwera, a następnie czy minął już czas wyświetlania danej wiadomości i czy rotator wykorzystał już wszystkie dostępne w danej fazie elementy (ma powrócić do początku). Następnie sprawdzana jest czy dany element nie jest już przeterminowany bądź nie jest zbyt szybko by go wyświetlić.

```

54 function rotator(){
55     if(typeof currentJson.responseJSON != 'undefined'){
56         //left-rotator
57         if(currentLeftTimer >= JSON.stringify(Number(
            currentJson['responseJSON']['left'][leftN]['duration']))) {

```

```

58     console.log('tu wlazlem');
59     if(leftN >= Object.keys(
        currentJson['responseJSON']['left']).length-1){
60         leftN = 0;
61     }
62     else{
63         console.log('dlugosc ok')
64         leftN++;
65     }
66     while(expired(
        currentJson['responseJSON']['left'][leftN]['since'],
        currentJson['responseJSON']['left'][leftN]['until'])) {
67         if(leftN >= Object.keys(
            currentJson['responseJSON']['left']).length-1){
68             leftN = 0;
69         }
70         else{
71             leftN++;
72         }
73     }

```

Listing 5: Początek funkcji rotatora napisanej w JavaScriptcie.

Instrukcja użytkowania aplikacji

Opis instalacji

W następnych podrozdziałach przedstawiony zostanie opis instalacji poszczególnych elementów systemu, który mógłby prowadzić do jego wdrożenia.

UWAGA: System w formie bez zabezpieczeń (m.in. uwierzytelniania użytkowników panelu) najlepiej instalować w wirtualnej sieci lokalnej (VLAN)^[22], w której znajdują się tylko urządzenia wyświetlające oraz zasób administrujący. W przeciwnym przypadku dostęp do managera i bazy danych będzie możliwy dla osób postronnych. Instrukcja przedstawia uruchomienie serwera w wersji deweloperskiej.

Instalacja serwera

Opis instalacji przygotowano dla systemu operacyjnego Ubuntu w wersji 17.10.

Pobranie, instalacja bazy MongoDB

1. Wykonać aktualizację bibliotek poleceniem `sudo apt-get update`;
2. Poleceniem `sudo apt-get upgrade -y` aktualizować system. Opcja `-y` służy do zainstalowania kolejnych modułów bez pytania o potwierdzenie;
3. Przy użyciu narzędzia Aptitude zainstalować serwer komendą `sudo apt-get install mongodb`;
4. Uruchomić serwer poleceniem `sudo systemctl start mongodb`. System poprosi o podanie hasła użytkownika;

Dodanie bazy systemu i kolekcji w bazie

1. Po zakończeniu instalacji serwera MongoDB uruchomić MongoShell poleceniem `mongo`;
2. Utworzyć bazę danych komendą `use infostradka`, powinien zostać wyświetlony komunikat `switched to infostradka`;
3. Utworzyć pięć kolekcji komendami:

```
1 db.createCollection("left")
2 db.createCollection("right")
3 db.createCollection("news")
4 db.createCollection("files")
5 db.createCollection("scrshots")
```

Listing 6: Kolejne komendy wpisywane celem utworzenia kolekcji na serwerze bazodanowym.

4. Sprawdzić poprawność utworzenia kolekcji poleceniem `show collections`;
5. Zamknąć konsolę MongoShell kombinacją klawiszy `Ctrl+C` lub poleceniem `exit`.

Pobieranie zawartości repozytorium, instalacja bibliotek i uruchomienie serwera

1. W przypadku braku gita, zainstalować go poleceniem `sudo apt install git`;
2. Pobrać kod z repozytorium przy użyciu komendy `git clone https://github.com/MateuszGrabuszynski/ict-infostradka.git`;
3. Przejść do folderu `infostradka-server` znajdującego się w folderze z projektem (`cd ict-infostradka/infostradka-server`);
4. W przypadku braku `python3-pip` zainstalować go poleceniem `sudo apt install python3-pip`;
5. Zainstalować odpowiednie biblioteki poleceniem `pip3 install -r requirements.txt`;
6. Uruchomić serwer Flask:

W terminalu (opcja A): `python3 -m swagger_server`;

W tle (opcja B): `nohup python3 -m swagger_server &`. Zapamiętać koniecznie numer PID procesu wyświetlony zaraz po uruchomieniu, np. [1] 18153. Logi zostaną zapisane w pliku tekstowym `nohup.out` w katalogu, z którego uruchomiony został serwer.

Zatrzymywanie serwerów

Serwer Flask pracujący w pierwszym planie (w terminalu, opcja A) zatrzymać można kombinacją klawiszy `Ctrl+C`. Uruchomiony w tle (opcja B) należy "zabić"

poleceniem `kill`, sprawdzając uprzednio numer procesu (polecenie `ps -a`), zgodnie z przykładem będzie to `kill 18153`.

Serwer bazy danych MongoDB można zatrzymać poleceniem `sudo systemctl stop mongod`.

Instalacja klienta

Przygotowanie Raspberry Pi polega na wykonaniu następujących kroków:

- 1) Pobrać ze strony <https://www.raspberrypi.org/downloads/raspbian/> pełną wersję systemu Raspbian (with desktop);
- 2) Wgrać pobrany obraz systemu na kartę microSD przy użyciu odpowiedniego programu (np. Etcher dostępny na <https://etcher.io/>);
- 3) Włożyć kartę pamięci do komputera Raspberry Pi;
- 4) Podłączyć do niego ekran i tymczasowo klawiaturę (oraz mysz w przypadku wyboru sieci Wi-Fi jako sieci dostępowej);
- 5) Uruchomić Raspberry Pi i poczekać na załadowanie pulpitu;
- 6) Podłączyć się do sieci Wi-Fi (ikona w prawym górnym rogu ekranu) bądź sprawdzić połączenie kablem;
- 7) Otworzyć terminal skrótem klawiszowym `Ctrl+Alt+T`;
- 8) Wpisać w terminalu `sudo raspi-config`;
 - a) W zakładce 5 (Interfacing Options) przejść do P2 (SSH) i uruchomić obsługę protokołu. Pomoże to w późniejszej konfiguracji ustawień;
 - b) W zakładce 7 (Advanced Options) przejść do A2 (Overscan) i wyłączyć tę opcję;
 - c) Wyjść z menu klawiszem `Esc`;
- 9) Aby zmienić hasło użytkownika wywołać `sudo passwd pi` a następnie podać hasło `raspberrypi` i dwukrotnie nowe hasło.

UWAGA: Użycie trywialnego hasła jest poważnym naruszeniem zasad bezpieczeństwa i może prowadzić do nieuprawnionego dostępu do urządzenia przez osoby trzecie.

- 10) W pliku `/etc/apt/source.list` należy odkomentować (usunąć znak hasza) jeśli widnieje jako zakomentowane, bądź dodać następujące repozytorium
`deb-src http://raspbian.raspberrypi.org/raspbian/ stretch main contrib non-free rpi`
- 11) Wpisać `sudo apt-get update` i poczekać na zakończenie pobierania;
- 12) Wpisać `sudo apt-get upgrade -y` i poczekać na zakończenie instalacji;
- 13) Zainstalować odpowiednie programy poleceniem `sudo apt-get install chromium-browser x11-xserver-utils unclutter`. Może pojawić się monit o przerwaniu instalacji niektórych programów ze względu na to iż są już zainstalowane, należy to zignorować;

- 14) Przygotować odpowiedni plik autostartu:
- Wpisać `nano ~/.config/lxsession/LXDE-pi/autostart`;
 - Zmienić zawartość pliku na następującą:

```
1 @lxpanel --profile LXDE-pi
2 @pcmanfm --desktop --profile LXDE-pi
3 @xset s off
4 @xset -dpms
5 @xset s noblank
6 @sed -i 's/"exited_cleanly": false/"exited_cleanly":
   true/' ~/.config/chromium Default/Preferences
7 @chromium-browser --noerrdialogs --kiosk --incognito
   --disable-translate http://[adres_tutaj]/v1/display
8 @/home/pi/screenshoter.sh [nazwa][adres_tutaj] &
9
10 @xsscreensaver -no-splash
11 @point-rpi
```

Listing 7: Fragment pliku autostartu.

UWAGA: [adres_tutaj] zamienić na odpowiedni adres serwera. [nazwa] zastąpić przypisaną urządzeniu nazwą.

Plik może zostać pobrany z repozytorium i umieszczony od razu we właściwym katalogu poprzez wykonanie kolejno poleceń:

```
cd /home/pi/.config/lxsession/LXDE-pi oraz
sudo wget https://raw.githubusercontent.com/MateuszGrabuszynski/ict-infostradka/master/autostart.
```

- 15) Przejść do ścieżki `/home/pi` poleceniem `cd /home/pi`;
- 16) Pobrać z serwera plik `screenshoter.sh` poleceniem
`wget https://raw.githubusercontent.com/MateuszGrabuszynski/ict-infostradka/master/screenshoter.sh`;
- 17) Zrestartować urządzenie poleceniem `sudo reboot now`;

Po zrestartowaniu urządzenia na monitorze powinna się wyświetlić odpowiednia strona. Pierwsze pobieranie paczek informacji z serwera może trwać kilka minut. Jeśli jednak po kilku minutach urządzenie nadal wyświetla biały ekran bądź wyświetli się komunikat "Aw, snap!" z dinozaurem - należy w pierwszej kolejności sprawdzić połączenie internetowe (polecenia `ifconfig` oraz `ping`), a następnie podane URL serwera i dane konfiguracyjne w pliku.

Opis użytkowania panelu administracyjnego

Panel administracyjny, do którego można dostać się poprzez endpoint `/v1/manager` składa się ze strony głównej oraz pięciu podstron. Odnośnik umożliwiający powrót z podstron do strony głównej znajduje się w górnym szarym pasku. Wciśnięcie, znajdującego się na dole, przycisku Github na którejkolwiek

z podstron przeniesie nas do repozytorium projektu, zaś przycisk Autorzy projektu pozwoli na przejście do sekcji pliku readme.md, w której znajdują się nazwiska autorów.

Strona główna

infostradka

Witaj!

- [Wyświetlacze](#)
- [Ustawienia sekcji głównej](#)
- [Ustawienia sekcji prawej](#)
- [Ustawienia paska informacyjnego](#)
- [Zarządzanie plikami](#)

© 2018 infostradka [Github](#) [Autorzy projektu](#)

Rysunek 7: Wygląd głównego okna managera

Strona główna managera zawiera odwołania do konkretnych części systemu zarządzającego. Naciśnięcie odnośnika Wyświetlacze powoduje przejście do okna Zrzuty ekranu, w którym można znaleźć podgląd z aktualnie działających wyświetlaczy. Kliknięcie w link Ustawienia sekcji głównej powoduje przejście do okna o tej samej nazwie, w której możliwe jest ustawienie zawartości głównego, największego okna. Wybranie Ustawienia sekcji prawej pozwala dostosować treść wyświetlaną w prawym, wąskim oknie. Odnośniki Ustawienia paska informacyjnego pozwala modyfikować tekstową zawartość dolnego paska. Zarządzanie plikami przenosi do paneli dotyczących dodawania oraz usuwania plików, które następnie będzie można używać w systemie.

Zrzuty ekranu



Rysunek 8: Wygląd okna Zrzuty ekranu

Podstrona znajduje się pod endpointem `/v1/manager/displays`. Można na niej wyświetlić zrzuty ekranów urządzeń podłączonych do systemu (tych, na których uruchomiono skrypt tworzący i przesyłający zrzuty). Obrazy z kolejnych urządzeń pojawiają się kolejno pod sobą. W przypadku, gdy do systemu nie są podpięte żadne urządzenia, strona pozostaje pusta.

Ustawienia sekcji głównej i prawej



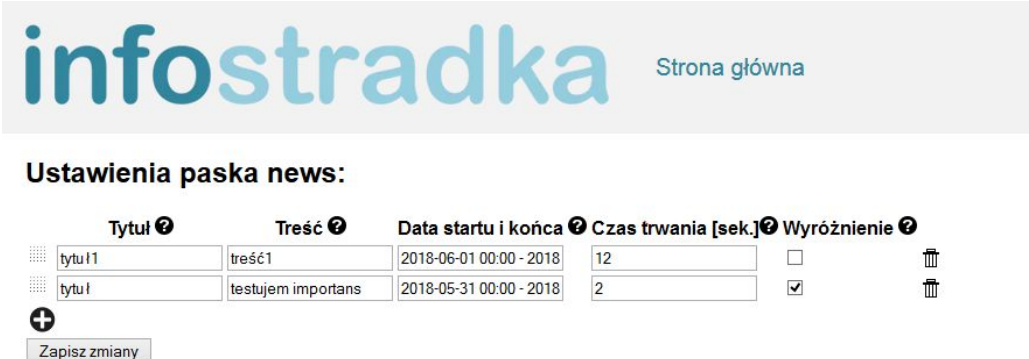
Rysunek 9: Po lewej stronie znajduje się okno ustawień dla sekcji głównej.

Po prawej stronie znajduje się okno ustawień dla sekcji prawej.

W oknach tych można dodawać nowe oraz edytować już istniejące elementy wyświetlane w rotatorach odpowiadających za sekcję główną i prawą. Kolejność elementów można zmienić poprzez chwycenie obrazka znajdującego się po lewej stronie wiersza, przeciągnięcie i upuszczenie w innym miejscu listy. Dla każdego

elementu można ustawić: rodzaj wyświetlanej treści, jego źródło, datę rozpoczęcia i zakończenia jej projekcji oraz okres pojedynczego wyświetlenia. Istniejący element można usunąć poprzez kliknięcie ikony kosza znajdującej się po jego prawej stronie. Dodanie nowego elementu do listy następuje po kliknięciu ikony plusa ustawionej poniżej listy elementów. W celu zapisania wprowadzonych modyfikacji należy wcisnąć przycisk Zapisz zmiany. Po zapisaniu wyświetlony zostaje odpowiedni alert.

Ustawienia paska news



Tytuł	Treść	Data startu i końca	Czas trwania [sek.]	Wyróżnienie
tytuł1	treść1	2018-06-01 00:00 - 2018	12	<input type="checkbox"/>
tytuł	testujem importans	2018-05-31 00:00 - 2018	2	<input checked="" type="checkbox"/>

[+](#)
Zapisz zmiany

Rysunek 10: Widok okna dla ustawień paska news.

Podobnie jak w przypadku zarządzania innymi sekcjami, elementy prezentowane są w postaci listy. Inny jest natomiast typ wyświetlanej treści, wiadomości tekstowe – stąd podmieniono odpowiednie kolumny na Tytuł i Treść, odpowiadające kolejno za większy i mniejszy napis na dolnym pasku informacyjnym. Możliwe jest ustawienie okresu w jakim wiadomość ma być wyświetlana. Dodano także kolumnę wyróżnienie, zmieniającą tło komunikatu podczas jego projekcji.

Panel zarządzania plikami



Pliki:

[Browse...](#) No files selected. [Dodaj pliki](#)

raspberry-pi-pinout.pr
1b36745-e2a7-4274-9

rpi-ha-paths.jpg
d44c6c4a-2cb8-4b97-

rpi-ha-all.jpg
3c34da17-1769-406b-

rpi-ha-db.PNG
db0bc9c2-caed-47b6

14233130_673976729-526c5e1a-eb6b-4bdf

14233130_673976729-3d2df113-f150-4447-89

13934966_660418374-d39a9bae-5a3b-4512

14233130_673976729-7a21c607-d47d-4052-

© 2018 infostradka [Github](#) [Autorzy projektu](#)

Rysunek 11: Wygląd okna zarządzania plikami.

W górnej części białego kontenera znajduje się wyszukiwarka plików na systemie użytkownika, pozwalająca na dodanie plików poprzez kliknięcie przycisku *Dodaj pliki*. Poniżej wyświetlają się pliki dodane przez użytkowników panelu, które to można używać na rotatorach jako pliki statyczne (niepobierane z Internetu). Pliki są identyfikowane poprzez nazwę oraz hash pliku. W zależności od typu dodanego pliku, zmienia się wyświetlana przy nim ikona. Kliknięcie ikonki kosza usuwa dany obraz z serwera. Każde takie usunięcie powinno być przemyślane – usunięcie używanych plików może spowodować problemy z wyświetlaniem rotatorów.

Napotkane problemy

No 'Access-Control-Allow-Origin' is present (...)

Podczas tworzenia oprogramowania klienta napotkano na problem związany z ładowaniem dokumentu JSON z zasobów zewnętrznego serwera. Cały komunikat wyświetlany w konsoli przeglądarki, brzmiał "Failed to load <adres do zasobu>/api: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access" ("Nie udało się załadować <adres do zasobu>/api: Brak nagłówka 'Access-Control-Allow-Origin' w żądanym zasobie. Nie jest możliwe pobranie danych pochodzących z 'null'").

Początkowo próbowano poprawić problem dodając do nagłówków odpowiedni przełącznik, jednak takie podejście nie zakończyło się powodzeniem. Następnie użyto rozszerzenia do przeglądarki, lecz ustalono, że może to znacznie pogorszyć bezpieczeństwo. Mogłoby to być także utrudnienie podczas instalacji systemu na urządzeniu klienta, zwłaszcza gdy ma ona przebiegać bez użycia myszy.

Ostatecznie problem został rozwiązany poprzez zmianę podejścia wyświetlania strony WWW wyświetlacza. Zamiast ładować ją z pamięci urządzenia, jest ona pobierana z serwera. Dzięki zastosowaniu takiego podejścia, w późniejszym etapie, można rozważyć dodanie do managera sposobu na dynamiczną edycję szablonu tablic informacyjnych w sposób scentralizowany, bez konieczności wgrywania dodatkowych plików na wszystkie urządzenia

Działo, ale przestało... (i nie wiadomo dlaczego)

Ze względu na to, iż zostawiliśmy sobie zbyt mało czasu, część kodów nie przechodziła porządnego code review, stąd wdarł się prosty błąd. W bazie danych zamieniono przypadkowo wprowadzanie liczby, na wprowadzanie tekstu, co popsuło działanie rotatorów na kilka dni przed oddaniem projektu.

Napisany JavaScript próbował przyrównać ciąg znaków do liczby, przez co zwracany wynik był zawsze nieprawdziwy. Pomogła zmiana wartości na numer poprzez funkcję `Number()`.

Wnioski i przemyślenia z pracy nad systemem

Tworzenie elementów projektu było zadaniem szczególnie rozwijającym. Nauczyliśmy się, że organizacja pracy, jest kluczowym elementem, warunkującym powodzenie jego zakończenia z zadowalającym wynikiem. Używane rozwiązania (architektura systemu, schemat bazy danych itp.) powinny być przemyślane, dokładnie rozrysowane bądź opisane na samym początku projektu, a koncepcja wytwarzanego oprogramowania wytłumaczona wszystkim członkom projektu.

Utworzenie odpowiedniego harmonogramu i trzymanie się terminów w nim zawartych jest bardzo ważne, patrząc pod kątem poziomu stresu i zmęczenia występującego w okolicach deadline'u. Człowiek niewyspany to człowiek zły, stąd niepotrzebnie zaczyna też rozmyślać nad zakończeniem swojego żywota, zwłaszcza gdy do terminu ostatecznego zostały trzy dni, a "ten string wcześniej był intem i działało, a teraz poświęciłem sześć godzin żeby znaleźć ten błąd". Pisanie trzech czwartych kodu w ostatnie dwa tygodnie było nieprzyjemne i na pewno odbiło się na jego jakości, lecz okazało się być rzeczą konieczną ze względu na to, iż wcześniej nie dotrzymywaliśmy części założonych terminów. Najgorszym z pomysłów było jednakże pozostawienie pisania tej dokumentacji na ostatnią chwilę.

Przyglądając się innym grupom doszliśmy też do wniosku, że dobrą praktyką wydaje się być wyznaczenie lidera, który kontroluje działania innych członków zespołu i wyznacza ewentualne zadania dodatkowe na bieżąco.

Warto jest doskonalić swoje umiejętności, także w rzeczach pozornie błahych i trywialnych, takich jak obsługa repozytoriów. Zdarzyło się (na szczęście tylko raz), wprowadzenie błędnego merge'a, którego trzeba było potem cofać. Unikanie tego typu problemów wymaga obycia z systemem, czyli zwyczajnie używania wszystkich oferowanych przez niego funkcji, czym w gicie na pewno nie są pojedyncze pushe na mastera.

Dobłą praktyką jest także korzystanie z dokumentacji używanych elementów stosu technologicznego. Dzięki nim dowiadujemy się o, nieznanych wcześniej, zaimplementowanych w nich funkcjach.

Rozbudowa systemu

Zabezpieczenia

W obecnej formie serwer nie posiada żadnych zabezpieczeń. Należałoby w pierwszej kolejności wprowadzić autoryzację użytkowników panelu zarządzającego. Najlepszym rozwiązaniem wydają się być Basic access authentication^[23] bądź JSON Web Token (JWT)^[24]. Opcjonalnie, można także dodać mechanizm kluczy API, który pozwoliłby na zablokowanie wyświetlania ekranu z rotatorami osobom nieuprawnionym w sieciach otwartych bez konieczności przeprowadzania konsultacji z administratorem.

Odporność na braki sieci

Podczas pisania dokumentacji zrozumiano błąd, który doprowadził do wystąpienia zjawiska zatrzymania rotatora w miejscu ostatnio wyświetlanego elementu, w przypadku rozłączenia z siecią. Należy zaimplementować mechanizm tymczasowego umieszczania pobieranego dokumentu JSON w zmiennej innej niż `currentJson`.

Walidacja pól w managerze

Część pól managera nie jest walidowana bądź jest walidowana częściowo. Na przykład w przypadku wpisania w pole Czas trwania niepoprawnej wartości, użytkownik nie otrzyma komunikatu o błędzie. Podobnie po wpisaniu w pole Wyświetlana treść całego linku do wideo z Youtube, użytkownik nie zostanie poinformowany o błędzie, a sam link nie zostanie poprawiony. Podobna sytuacja występuje w przypadku dodawania plików. Sprawdzane są jedynie informacje o typie pliku, czy nie jest on pusty i czy jego nazwa nie jest pusta. Bardzo rzadko, jednak zdarza się, że pliki nie zostaną poprawnie dodane do katalogów serwera, a znajdują się w bazie danych.

Predefiniowane elementy rotatora

Propozycją rozbudowy jest dodanie predefiniowanych paneli rotatorów. Użytkownik przebywający w managerze, w zakładce Ustawienia sekcji prawej bądź Ustawienia sekcji lewej miałby możliwość wybrania rodzaju treści jako jeden z predefiniowanych elementów, którego treść dostosowywałaby się do treści pola Wyświetlana treść. Na przykład w przypadku wybrania panelu Pogoda i wpisania miasta Poznań, wyświetlana byłaby pogoda z Poznania. Ciekawym pomysłem mogłoby być także ustawienie w tych polach informacji przydatnych

dla dojeżdżających studentów i wykładowców – np. liczby rowerów w stojaku PRM Centrum Wykładowe.

Składanie propozycji poprawek

Aby przyłączyć się do prac nad projektem można użyć mechanizmu forkowania^[25] repozytorium, a po wprowadzeniu zmian złożyć *Pull Request* (PR). Należy zamieścić dokładny opis zmian nie pozostawiający wątpliwości odnośnie przeprowadzonych edycji. Uprasza się, aby podczas dodawania kodu pamiętać o prawach autorskich i majątkowych osób trzecich oraz zwracać szczególną uwagę na licencje dodawanych bibliotek czy wtyczek. Członkowie zespołu sprawdzą i przedyskutują zmiany, a jeśli będą one sensowne, dodadzą je do projektu.

Przypisy końcowe

- [1] <https://www4.put.poznan.pl/pl/media/system-informacji-wewnetrznej>
System Informacji Wewnętrznej (dostęp 3.06.2018)
- [2] <https://www.amazon.com/Amazon-Echo-Show-Alexa-Enabled-Black/dp/B01J24C0TI> (j.ang.)
Echo Show | Alexa-enabled Bluetooth Speaker with 7" Screen - Black (dostęp 3.06.2018)
- [3] https://www.amazon.com/dp/B073SQYXTW/ref=fs_ods_rk (j.ang.)
Echo Spot | Alexa-enabled Bluetooth Speaker with 2.5" Screen - Black (dostęp 3.06.2018)
- [4] <https://www.youtube.com/watch?v=ogfYd705cRs> (j.ang.)
Keynote (Google I/O '18), wideo od 28:10 do 31:17 (dostęp 3.06.2018)
- [5] <https://www.twowaymirrors.com/smart-mirror/> (j.ang.)
Smart Mirror Store | Optical Quality Beamsplitter Glass (dostęp 3.06.2018)
- [6] <http://www.gazetawroclawska.pl/komunikacja/mpk/a/uruchomiono-nowe-kasowniki-jak-ku-pic-bilet.12971366/>
Uruchomiono nowe kasowniki. Jak kupić bilet? - Gazetawroclawska.pl (dostęp 3.06.2018)
- [7] http://www.bryston.com/products/digital_audio/BDP-Pi.html (j.ang.)
BDP-Pi Digital Player - Bryston (dostęp 3.06.2018)
- [8] <https://pl.aliexpress.com/item/Bitcon-Miner-1TH-A1-28nm-Asic-Miner-1000GH-Dragon-Miner-Super-Btc-Miner-for-Bitcoin-Mining/32222774791.html> (j. ang.)
Bitcon Górnik LKETC 1TH A1 28nm (...) (dostęp 3.06.2018)
- [9] <http://flask.pocoo.org/docs/1.0/license/> (j. ang.)
Licence — Flask 1.0.2 documentation (dostęp 3.06.2018)
- [10] <https://www.chromium.org/> (j. ang.)
The Chromium Projects (dostęp 3.06.2018)
- [11] <https://en.wikipedia.org/wiki/Scrot> (j. ang.)
Scrot - Wikipedia (dostęp 6.06.2018)
- [12] <https://blog.strefakursow.pl/co-jest-lesssass-jak-moge-wykorzystac-w-praktyce/>
Co to jest LESS/SASS i jak mogę to wykorzystać w praktyce? | Programowanie (dostęp 7.06.2018)
- [13] <https://github.com/isocra/TableDnD> (j. ang.)
isocra/TableDnD: jQuery plug-in to drag and drop rows in HTML tables (dostęp 5.06.2018)

- [14] www.daterangepicker.com
Date Range Picker — JavaScript Date & Time Picker Library
- [15] <https://github.com/ticktoo/simpleClock> (j. ang.)
ticktoo/simpleClock: A tiny and simple jQuery Plugin to create an animated clock with date and time. (dostęp 7.06.2018)
- [16] <https://www.jetbrains.com/pycharm/download> (j. ang.)
Download PyCharm: Python IDE for Professional Developers by JetBrains (dostęp 7.06.2018)
- [17] <https://git-scm.com/download> (j. ang.)
Git - Downloads (dostęp 7.06.2018)
- [18] <https://www.putty.org/>
Download PuTTY - a free SSH and telnet client for Windows
- [19] <https://www.virtualbox.org/wiki/Downloads>
Downloads – Oracle VM VirtualBox
- [20] <http://releases.ubuntu.com/17.10/> (j. ang.)
Ubuntu 17.10.1 (Artful Aardvark) (dostęp 7.06.2018)
- [21] <https://pl.wikipedia.org/wiki/WebSocket>
Websocket – Wikipedia, wolna encyklopedia (dostęp 7.06.2018)
- [22] https://pl.wikipedia.org/wiki/Wirtualna_sie%C4%87_lokalna
Wirtualna sieć lokalna – Wikipedia, wolna encyklopedia
- [23] https://en.wikipedia.org/wiki/Basic_access_authentication (j. ang.)
Basic access authentication - Wikipedia (dostęp 7.06.2018)
- [24] https://en.wikipedia.org/wiki/JSON_Web_Token (j. ang.)
JSON Web Token - Wikipedia (dostęp 7.06.2018)
- [25] <https://help.github.com/articles/fork-a-repo/> (j. ang.)
Fork a repo - User Documentation (dostęp 7.06.2018)