Bezpečný testovací systém

Jozef Komáromy

April 30, 2024

1 Prehľad problému

Účelom tohto projektu je vytvoriť testovací systém pre školu, ktorý zabezpečí počítač testovaného študenta tak, aby nemohol podvádzať. Vo väčšine prípadov prebieha digitálne testovanie prostredníctvom web-aplikácie v prehliadači (napr. Google Forms, Moodle, a pod.). Tieto spôsoby testovania sú často nezabezpečené proti najjednoduchším spôsobom podvádzania. Testovaná osoba si napríklad môže jednoducho otvoriť ďalší tab a vyhladávať správne odpovede na Googli. Aby sme vyriešili problém podvádznia bez priameho zásahu do testovacieho prostredia, potrebujeme vyvinúť aplikáciu, ktorá nahradí úlohu tradičného webového prehliadaču pri testovaní. Testovacie prostredie teda bude spustené cez túto aplikáciu. Možnosti testovanej osoby sú obmedzené tak, že nemôže na počítači zobrazovať iné aplikácie, a prechádzať na web-stránky ktoré mu neboli explicitne povolené.

2 Riešenie

Mojím riešením tohto problému je sada softvéru pozostavájúca z 4 častí:

1. *Client*
2. *Vanguard*
3. *Server*
4. *Supervisor*

2.1 Client

Aplikácia spustená na počítači testovanej osoby (ďalej študent), v ktorej je spustené webové testovacie prostredie. Po otvorení aplikácie má študent možnosť pripojiť sa do miestnosti vytvorenej dohliadajúcou osobou (ďalej učiteľ). Po pripojení do miestnosti sa načíta konfigurácia pre túto miestnosť a zobrazí sa testovacie prostredie. Taktiež sa spustí program Vanguard, ktorý upozorňuje na podozrivé správanie študenta. Informácie o podozrivom správaní sa ďalej posielajú na Server.

2.2 Vanguard

Pomocný program pre testovaciu aplikáciu, ktorý sleduje akcie študenta vrámci celého operačného systému. Ak vyhodnotí že správanie študenta je podozrivé, posúva túto informáciu do aplikácie Client. Zahŕňa viacero služieb, ktoré sledujú napr. aké okno je v popredí systému.

2.3 Server

Softvér bežiaci na vzdialenom zariadení, slúži zväčša ako komunikačná vrstva medzi aplikáciami Client a Supervisor. Taktiež sa tu uchovávajú informácie o miestnostiach, a študentoch v nich. Poskytuje aj bezpečné prihlásenie učiteľa do systému, a pripojenie študenta do miestnosti.

2.4 Supervisor

Webová aplikácia ktorá učiteľovy poskytuje zobrazenie informácií získaných zo Serveru. Učiteľ sa prihlási do systému a následne môže vytvárať miestnosti, monitorovať študentov pripojených do miestnosti, a diaľkovo ovládať ich klientskú aplikáciu.

3 Špecifikácia

3.1 Client

3.1.1 Technológie

Aplikácia klienta je implementovaná v programovacom jazyku *TypeScript* (kompilovaný do *JavaScript*). Používa *UI* framework *Electron* a frontendový framework *Vue*. Pre komunikáciu so serverom je použitá knižnica *Axios*.

3.2 Vanguard

3.2.1 Technológie

Program Vanguard je implementovaný v programovacom jazyku *C* v prostredí *mingw64*. Používa knižnicu *winpthreads* (port pthreads.h pre OS Windows), a *Windows API*.

3.2.2 Komunikácia

Program je spustený ako podproces aplikácie klienta. Komunikácia s klientom sa realizuje posielaním binárny balíkov informácií na štandardnom vstupe a výstupe.

3.2.3 Služby

3.3 Server

3.3.1 Technológie

Server je implementovaný v programovacom jazyku *PHP* pre webový server *Apache* a databázu *MySQL/MariaDB*. Pre komunikáciu s databázou sa používa vstavaná *PHP* knižnica *PDO*. Pre zamedzenie prístupu neoprávneným requestom je použitý vstavaný *Apache* modul *mod_rewrite*.

Pre komunikáciu s aplikáciou klienta a applikáciou Supervisor sa používajú *POST* requesty cez protokol *HTTP(S)* kde klient a server posielajú dáta v formáte *JSON*.

3.3.2 Konfigurácia

Konfiguračný súbor pre webový server *Apache httpd.conf* je vytvorený zo šablónového súboru *httpd.tpl.conf* . Pri manuálnej konfigurácií treba do tohto súboru doplniť za *ROOT* absolutnu cestu ku koreňovému priečinku projektu v ktorom sa bude nachádzať *httpd.conf*, za *LOCATION* treba doplniť http cestu na ktorej bude rezidovať adresár */public*. Po doplnení informácií do súboru *httpd.conf* treba túto konfiguráciu zahrnúť do hlavnej konfigurácie *Apache*, napr.:

```
Include "/var/www/priscilla-server/httpd.conf"
```

httpd.conf - V súbore *httpd.conf* je nastavená premenná prostredia *SERVER_ROOT* na vyššie spomínanú cestu. Pokyn `php_value include_path "ROOT/vendor"` hovorí *PHP* interpreteru aby hladal importované knižnice v adresáry *vendor*. Pokyn `php_value auto_prepend_file "ROOT/vendor/autoload.php"` hovorí *PHP* interpreteru aby vždy najskôr spustil súbor *autoload.php* pred požadovaným php skriptom.

env.ini - Server sa konfiguruje pomocou súboru *env.ini* , v ktorom treba nastaviť nasledujúce premenné.

DATABASE_PROTO - druh databázy, netreba meniť pôvodnú hodnotu `"mysql"`

DATABASE_HOST - adresa zariadenia na ktorom sa nachádza databázový server

DATABASE_PORT - port na ktorom je databázový server spustený

DATABASE_NAME - názov databázy

DATABASE_USER - prihlasovacie meno používateľa databázy

DATABASE_PASSWORD - heslo používateľa databázy

Príklad nastavenia pre *XAMPP*:

```
DATABASE_PROTO="mysql"
DATABASE_HOST="localhost"
DATABASE_PORT="3306"
DATABASE_NAME="supervisor"
DATABASE_USER="root"
DATABASE_PASSWORD=""
```

autoload.php - skript načíta konfiguráciu zo súboru *env.ini*

3.3.3 Databáza

Databázová schéma sa nachádza v súbore */database/schema.sql*. Obsahuje nasledovné entity.

user (ďalej používateľ, učiteľ) - Reprezentuje učiteľa ktorý sa vie prihlásiť do systému, vytvárať miestnosti, sledovať a dialkovo ovládať klientov.

```
create table `user` (
    `id` int primary key auto_increment,
    `username` varchar(16) unique not null,
    `displayname` varchar(32) not null,

    `password_hash` varchar(64) not null,
    `password_salt` varchar(32) not null
);
```

username - prihlasovacie meno používateľa

displayname - meno používateľa, ktoré sa zobrazuje napríklad v názve miestnosti

password_salt - náhodne generovaný string na strane servera, ktorý sa pridáva k heslu pri hashovaní pre zvýšenie bezpečnosti

password_hash - zahashované (algo. *SHA256*) heslo + *password_salt* , používa sa na overenie používateľa pri prihlásení

session - Reprezentuje reláciu používateľa, ktorá vznikne prihlásením a zanikne odhlásením.

```
create table `session` (
    `id` varchar(32) primary key,
    `user_id` int not null,

    foreign key (`user_id`) references `user`(`id`)
);
```

id - Náhodne generovaný string s dĺžkou 32 znakov. Vytvori sa pri prihlásení používateľa. Používa sa pre overenie používateľa pri requestoch na *UserEndpoint*

room - Reprezentuje miestnosť v ktorej "prebieha" testovanie. Miestnosť vytvára používateľ. Do miestnosti sa pripájajú klienti pomocou *join_code* ktorý následne posielajú na server eventy ktoré budú v databáze prepojené s miestnosťou do ktorej patrí klient. Používateľ po vytvorení miestnosti môže odoberať eventy vytvorené klientami pomocou *watch_code*

```
create table `room` (
  `id` int primary key auto_increment,
  `name` varchar(32) not null,
  `join_code` varchar(16) not null,
  `watch_code` varchar(16) not null,
  `config` json not null,

  `owner_id` int not null,
  foreign key (`owner_id`) references `user`(`id`)
);
```

join_code - Náhodne generovaný string, dlhý 6 znakov. Vznikne pri vytvorení miestnosti. Učiteľ tento kód poskytne študentom aby sa mohli pripojiť do miestnosti.

watch_code - Náhodne generovaný string, dlhý 12 znakov. Vznikne pri vytvorení miestnosti. Používa sa pri overení oprávnenia používateľa sledovať miestnosť pri requestoch na *WatchEndpoint*

config - Konfigurácia aplikácie *Client* ktorá mu bude poskytnutá keď sa pripojí do miestnosti.

owner_id - *id* používateľa, ktorý vytvoril miestnosť

client - Reprezentuje študenta, ktorý sa pripojil do miestnosti. Treba podotknúť, že študent si v systéme nevytvára účet, pripája sa do každej miestnosti samostatne. Každý klient patrí iba jednej miestnosti.

```
create table `client` (
    `id` int primary key auto_increment,
    `ip_address` varchar(32) not null,
    `name` varchar(32) not null,
    `secret` varchar(32) not null,
    `state` json default('{}'),

    `room_id` int not null,
    foreign key (`room_id`) references `room`(`id`)
);
```

secret - Náhodne generovaný string, dlhý 32 znakov. Vznikne pri prihlásení klienta do miestnosti. Používa sa pri overení klienta pri requestoch na *ClientEndpoint*

state - Stav klienta. Periodicky sa aktualizuje cez endpoint */client/pushstate*.

client_event - Reprezentuje udalosť ktorá sa stala na strane klienta. Môže to byť napríklad oznámenie o tom, že študent porušil pravidlá testu, úspešne sa prihlásil do testovacieho prostredia, že už test dokončil, alebo to môže byť odpoveď na správu (*message*) od učiteľa.

```
create table `client_event` (
    `id` int primary key auto_increment,
    `created` datetime not null default(now()),
    `data` json not null,

    `client_id` int not null,
    `room_id` int not null,

    foreign key (`client_id`) references `client`(`id`),
    foreign key (`room_id`) references `room`(`id`)
);
```

data - podrobnosti eventu vo formáte *JSON*, napr.:

```
{
  "module": "Vanguard",
  "timestamp": "2024-04-27T03:26:34.848Z",
  "severity": 3,
  "message": "Wrong window in foreground",
  "data": {
    "handle": "0404010000000000",
    "title": "Google - Firefox"
  }
}
```

room_event - Reprezentuje udalosť ktorá sa týka miestnosti. Môže to byť napríklad oznámenie o tom, že sa študent pridal alebo odpojil z miestnosti.

```
create table `room_event` (
    `id` int primary key auto_increment,
    `created` datetime not null default(now()),
    `data` json not null,

    `client_id` int,
    `room_id` int not null,

    foreign key (`client_id`) references `client`(`id`),
    foreign key (`room_id`) references `room`(`id`)
);
```

client_id - *not null* pretože event sa nemusí týkať klienta. Napríklad ak to je oznámenie o vytvorení miestnosti.

data - podrobnosti eventu vo formáte *JSON*, napr.:

```
{ "message": "Client 'Jozef Komaromy' joined the room" }
{ "message": "Room created" }
```

message - Správa zo strany učiteľa pre klienta. Zvyčajne to je pokyn pre aplikáciu klienta, aby napríklad zablokovala prístup študentovy, alebo naopak odblokovala.

```
create table `message` (
    `id` int primary key auto_increment,
    `data` json not null,

    `client_id` int not null,

    foreign key (`client_id`) references `client`(`id`)
)
```

data - pokyny pre klientskú aplikáciu vo formáte *JSON*, napr.:

```
{ "action": "lock" }
{ "action": "unlock" }
{ "action": "clear_warning" }
```

3.3.4 Vendor

PHP skripty v adresáry */vendor* niesu prístupné verejnosti, slúžia ako knižnica pre endpointy.

autoload.php - Automaticky načítaný pre každý request na endpoint, načíta premenné prostredia zo súboru *env.ini*, a includne *util.php*.

Response.php (*interface*) - Reprezentuje odpoveď servera, obsahuje kód odpovede a dáta odpovede.

> **ResponseSuccess** - má kód odpovede 0, a dáta reprezentujúce výsledok operácie
>
> **ResponseError** - má kód odpovede 0, a dáta obsahujúce iba správu (*message*), ktorá popisuje chybu ktorá nastala

Database.php (*class*) - Pripojenie na databázu pomocou knižnice vstavanej *PHP* knižnice *PDO*, detaily pripojenia sú načítané zo súboru *env.ini*

Auth.php - Kryptografické funkcie na generovanie bezpečných hesiel, kontrola hesla, hashovanie, a taktiež interakcie s databázou týkajúce sa autentikácie používateľa.

Endpoint.php (*abstract class*) - Je abstrakciou pre koncový bod servera. Automaticky dekoduje request body z formátu *JSON* do php array. Taktiež zachytáva chyby, aby neboli viditelné pre klienta. Vždy vracia výsledok vo formáte *JSON*, ktorý vždu obsahuje hodnotu *code*, ktorá reprezentuje úspešnosť požiadavky. Ak *code = 0*, znamená to, že operácia prebehla úspešne, nenulové hodnoty znamenajú chybu. Ak nastala chyba jej popis sa nachádza v hodnote *message*.

```
// príklad výsledku úspešnej operácie
{
  "code": 0,
  // Výsledok úspešnej operácie
}

// príklad výsledku operácie s chybou
{
  "code": 1,
  "message": "Input argument username not provided"
}

// konvenčný výstup pri nezadaní požadovaných hodnôt pre vstupu
{
  "code": 1,
  "message": "Bad input"
}
```

UserEndpoint.php (*abstract class extends Endpoint*) - Endpoint ktorý najskôr vykoná autentikáciu používateľa, a ak vyhodnotí že je prihlásený, vykoná operáciu. V request body (*JSON*) musí byť poskytnutá hodnota *session*

```
// príklad vstupu
{ "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN", /* argumenty pre špecifický endpoint */ }

// výstup ak nebol poskytnutý session
{ "code": 1, "message": "Not logged in" }

// výstup ak je session nesprávny
{ "code": 1, "message": "Invalid session" }
```

WatchEndpoint.php (*abstract class extends UserEndpoint*) - Endpoint ktorý overuje prístup používateľa k miestnosti. Pre overenie musí poskytnúť hodnotu *watch_code* ktorá je jedinečná pre každú miestnosť v systéme. Taktiež je potrebné poskytnúť *session* pre overenie prihlasenia ako u *UserEndpoint*

```
// príklad vstupu
{
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN",
  "watch_code": "oSWnHmyKgz32",
  /* argumenty pre špecifický endpoint */
}

{ "code": 1, "message": "No watch code" } // výstup ak nebol poskytnutý watch_code
{ "code": 1, "message": "Invalid watch code" } // výstup ak je watch_code nesprávny
```

ClientEndpoint.php (*abstract class extends Endpoint*) - Endpoint ktorý podobne ako *UserEndpoint* overí používateľa na základe parametru *secret*

```
// príklad vstupu
{ "secret": "VvJnJHmaderb", /* argumenty pre špecifický endpoint */ }

// výstup ak nebol poskytnutý secret
{ "code": 1, "message": "No secret provided" }

// výstup ak je secret nesprávny
{ "code": 1, "message": "Invalid secret" }
```

3.3.5 Endpointy

Endpointy sa nachádzajú v adresáry */public*. Sú rozdelené do podpriečinkov podľa obmedzenia prístupu. Pristup k endpointom je obmedzený súborom *.htaccess*, a taktiež sú presmerované adresy bez prípony *.php* na *PHP* skripty (*/client/joinroom* → */client/joinroom.php*)

/operator - endpointy dostupné iba pre administrátora serveru, prístup je obmedzený súborom *.htaccess* tak, aby mali prístup k tomuto priečinku iba requesty z adresy *127.0.0.1*

/operator/deploy (*extends Endpoint*) - Spustí na *MySQL/MariaDB* servery *SQL* skript */database/schema.sql* , ktorý vytvorí tabulky v databáze.

/operator/adduser (*extends Endpoint*) - Prida nového používateľa. Vstupom sú hodnoty *username*, *password* a *displayname* (ak nie je zadané, bude rovnaké ako *username*). Taktiež overuje či už používateľ s daným *username* v systéme už neexistuje.

```
// príklad vstupu
{
  "username": "jskalka",
  "password": "heslo123",
  "displayname": "Ján Skalka"
}

// výstup
{ "code": 0 } // používateľ bol pridaný
{ "code": 1, "message": "Bad password" } // heslo nie je dostatočne silné
{ "code": 1, "message": "User jskalka already exists" }
```

/info - verejne dostupné endpointy, ktoré poskytujú informácie o servery.

/info/features (*extends Endpoint*) - Pomôcka pri nastavovaní klienta, poskytuje informácie o nastavení rôznych služieb serveru, napr. akým spôsobom má klient posielať eventy.

```
// príklad výstupu
{
  "supervisor": {
    "protocol": "http"
  },
  "messages": {
    "protocol": "http-refresh"
  }
}
```

/user - Endpointy určené pre aplikáciu *Supervisor* . Tykajú sa autentikácie používateľa (*login*, *logout*, *info*), spravovania miestností (*createroom*, *getrooms*), sledovania udalostí týkajúcich sa miestnosti (*getclients*, *getevents*), a komunikácie s aplikáciou klienta (*message*).

/user/login (*extends Endpoint*) - Zabezpečuje prihlásenie používateľa podľa mena a hesla. Po úspešnom prihlásení vráti používateľovy session token ktorý použije pre overenie pri nasledujúcich requestoch. Session platí až do odhlásenia.

```
// príklad vstupu
{
  "username": "jskalka",
  "password": "heslo123"
}

// výstup ak prihlásenie bolo úspešné
{
  "code": 0,
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN"
}

{ "code": 1, "message": "User jskalka does not exist" } // ak používateľ s daným menom nie je regi
{ "code": 1, "message": "Wrong password" }
```

/user/logout (*extends UserEndpoint*) - Po overení prihlásenia (*UserEndpoint*) invaliduje session používateľa.

```
// príklad vstupu
{ "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN" }

{ "code": 0 } // výstup ak odhlásenie bolo úspešné
```

/user/info (*extends UserEndpoint*) - Ak je používateľ prihlásený, vráti informácie o používateľovy.

```
// príklad vstupu
{ "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN" }

{
  "code": 0,
  "id": 1,
  "username": "jskalka",
  "displayname": "Ján Skalka"
}
```

/user/createroom (*extends UserEndpoint*) - Vytvorí miestnosť podla parametrov *name* a *config* poskytnutých na vstupe. Parameter *config* je string obsahujúci *JSON* ktorý bude poskytnutý aplikácií *Client* pri pripojení do miestnosti. Ak nie je poskytnutý názov miestnosti, vytvorí sa z *displayname* používateľa ("Ján Skalka's room").

```
// príklad vstupu
{
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN",
  "name": "Python Test - sk. 1",
  "config": "{ ... }" // konfigurácia pre Client aplikáciu
}

// príklad výstupu
{
  "id": 1,
  "name": "Python Test - sk. 1",
  "join_code": "jxCzW8",
  "watch_code": "oSWnHmyKgz32"
}

// ak nebola poskytnutá konfigurácia
{ "code": 1, "message": "No config provided" }
```

/user/getrooms (*extends UserEndpoint*) - Ak je používateľ prihlásený, vráti informácie o miestnostiach ku ktorým má prístup.

```
// príklad vstupu
{ "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN" }

// príklad výstupu
{
  "code": 0,
  "rooms": [
    {
      "id": 1,
      "name": "Python Test - sk. 1",
      "join_code": "jxCzW8",
      "watch_code": "oSWnHmyKgz32"
    },
    {
      "id": 4,
      "name": "JavaFX Test - sk. 2",
      "join_code": "2Gos11",
      "watch_code": "PwUZa1mkK9he"
    }
  ]
}
```

/user/getclients (*extends WatchEndpoint*) - Vráti informácie o klientoch pripojených do miestnosti.

```
// príklad vstupu
{
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN",
  "watch_code": "oSWnHmyKgz32",
}

// príklad výstupu
{
  "code": 0,
  "clients": [
    {
      "id": 1,
```

```
      "ip_address": "129.11.55.12",
      "name": "Jozef Komáromy",
      "state": "{ ... }" // JSON string, obsahuje stav klienta
    },
    {
      "id": 5,
      "ip_address": "129.11.55.13",
      "name": "Dominik Vrana",
      "state": "{ ... }"
    },
  ]
}
```

/user/getevents (*extends WatchEndpoint*) - Vráti eventy z tabuliek *room_event* a *client_event* pre sledovanú miestnosť, filtrované podľa parametrov requestu.

client_id - obmedzuje požiadavku iba na eventy týkajúce sa daného klienta

last_room_event_id - určuje posledné *id* udalosti miestnosti ktorú už používateľ má. Ak je jeho hodnota *-1*, vrátia sa všetky eventy. Ak nie je tento parameter poskytnutý, nevrátia sa žiadne eventy miestnosti.

last_client_event_id - určuje posledné *id* udalosti klienta ktorú už používateľ má. Ak je jeho hodnota *-1*, vrátia sa všetky eventy. Ak nie je tento parameter poskytnutý, nevrátia sa žiadne eventy klienta.

```
// príklad vstupu
{
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN",
  "watch_code": "oSWnHmyKgz32",
  "last_room_event_id": 20,
  "last_client_event_id": 92,
  "client_id": 5
}

// príklad výstupu
{
  "code": 0,
  "client_events": [
    {
      "id": 102,
      "created": "2024-04-27 15:06:07"
      "data": "{ ... }" // JSON string obsahujuci podrobnosti o evente
      "client_id": 5,
      "room_id": 2
    },
    {
      "id": 110,
      "created": "2024-04-27 15:06:23"
      "data": "{ ... }"
      "client_id": 5,
      "room_id": 2
    }
  ],
  "room_events": [
    {
      "id": 22,
```

```
      "created": "2024-04-27 15:06:55"
      "data": "{ ... }"
      "client_id": 5,
      "room_id": 2
    }
  ]
}
```

/user/message (*extends UserEndpoint*) - Pošle správu klientovy. Správa pre klienta je poskytnutá ako *JSON* string v parametri *data*. Taktiež je potrebné poskytnúť id klienta pre ktorého je správa určená v *client_id*

```
// príklad vstupu
{
  "session": "QWBUP9WcuBt6zVjPo4YOwG3d4SLBzehN",
  "client_id": 2,
  "data": "{ ... }" // JSON string obsahujúci správu
}

{ "code": 0 } // výstup ak poslanie správy prebehlo úspešne
{ "code": 1, "message": "Bad input" } // ak nebol poskytnutý požadovaný parameter
```

Príklad komunikácie s klientom:

1. klient zmení stav na `{ "locked": true }`
2. supervisor sa o tomto dozvie po načítaní stavu klienta cez *getclients*
3. odošle klientovy správu `{ "action": "unlock" }`
4. klient načíta a spracuje správu
5. klient zmení stav na `{ "locked": false }`
6. u supervisora sa zobrazí zmena po načítaní stavu klienta

/client - Endpointy cez ktoré komunikuje aplikácia *Client* so serverom, a učiteľom.

/client/joinroom (extends Endpoint) - Pripojenie klienta do miestnosti. Parameter *name* je meno študenta. Parameter *join_code* je prihlasovací kód ktorý študent fyzicky obdržal od učiteľa. Podľa *join_code* sa vyhľadá miestnosť a študent bude do nej pridaný. Vracia *secret* - token ktorým klient preukazuje svoju totožnosť pri ďalších requestoch na server, *room_name* - názov miestnosti, a *config* - string obsahujúci konfiguráciu pre klienta v formáte *JSON*.

```
// príklad vstupu
{
  "name": "Jozef Komaromy",
  "join_code": "jxCzW8"
}

// príklad výstupu
{
  "secret": "0aUHzmL4YgE0",
  "room_name": "Python Test - sk. 1",
  "config": "{ ... }" // JSON string obsahujúci konfiguráciu klienta
}
```

/client/pushstate (extends ClientEndpoint) - Klient posiela na server svoj aktuálny stav. Parameter *state* je string obsahujúci stav clienta vo formáte *JSON*

```
// príklad vstupu
{
  "secret": "0aUHzmL4YgE0",
  "state": "{ ... }" // JSON string obsahujúci stav klienta
}

{ "code": 0 } // ak bol stav aktualizovany úspešne
{ "code": 1, "message": "Bad input" } // ak nebol poskytnutý parameter state
```

/client/pushevent (extends ClientEndpoint) - Klient posiela na server event. Parameter *data* je string obsahujúci event vo formáte *JSON*

```
// príklad vstupu
{
  "secret": "0aUHzmL4YgE0",
  "data": "{ ... }" // JSON string obsahujúci event
}

{ "code": 0 } // ak bol event pridaný do databázy úspešne
{ "code": 1, "message": "Bad input" } // ak nebol poskytnutý parameter data
```

/client/getmessage - Vráti klientovy správy jemu adresované. Parameter *last_id* je *id* poslednej správy, ktorú má klient. Ak *last_id* = *-1*, server vráti všetky správy pre klienta.

```
// príklad vstupu
{
  "secret": "0aUHzmL4YgE0",
  "last_id": 18
}

// príklad výstupu
{
  "code": 0,
  "messages": [
    {
      "id": 20,
      "client_id": 2,
      "data": "{ ... }" // správa vo formáte JSON
    },
    {
      "id": 25,
      "client_id": 2,
      "data": "{ ... }"
    }
  ]
}

{ "code": 1, "message": "Bad input" } // ak nebol poskytnutý parameter last_id
```

3.4 Supervisor

3.4.1 Technológie

Aplikácia *Supervisor* je webová aplikácia implementovaná v programovacom jazyku *TypeScript* (kompilovaný do *JavaScript*). Používa frontendový framework *Vue*. Pre komunikáciu so serverom je použitá knižnica *Axios*.