Stargate SG•1 Fanpage

•••

Súťaž pre deviatakov: Kategória Programovanie

Jakub Krčmárik

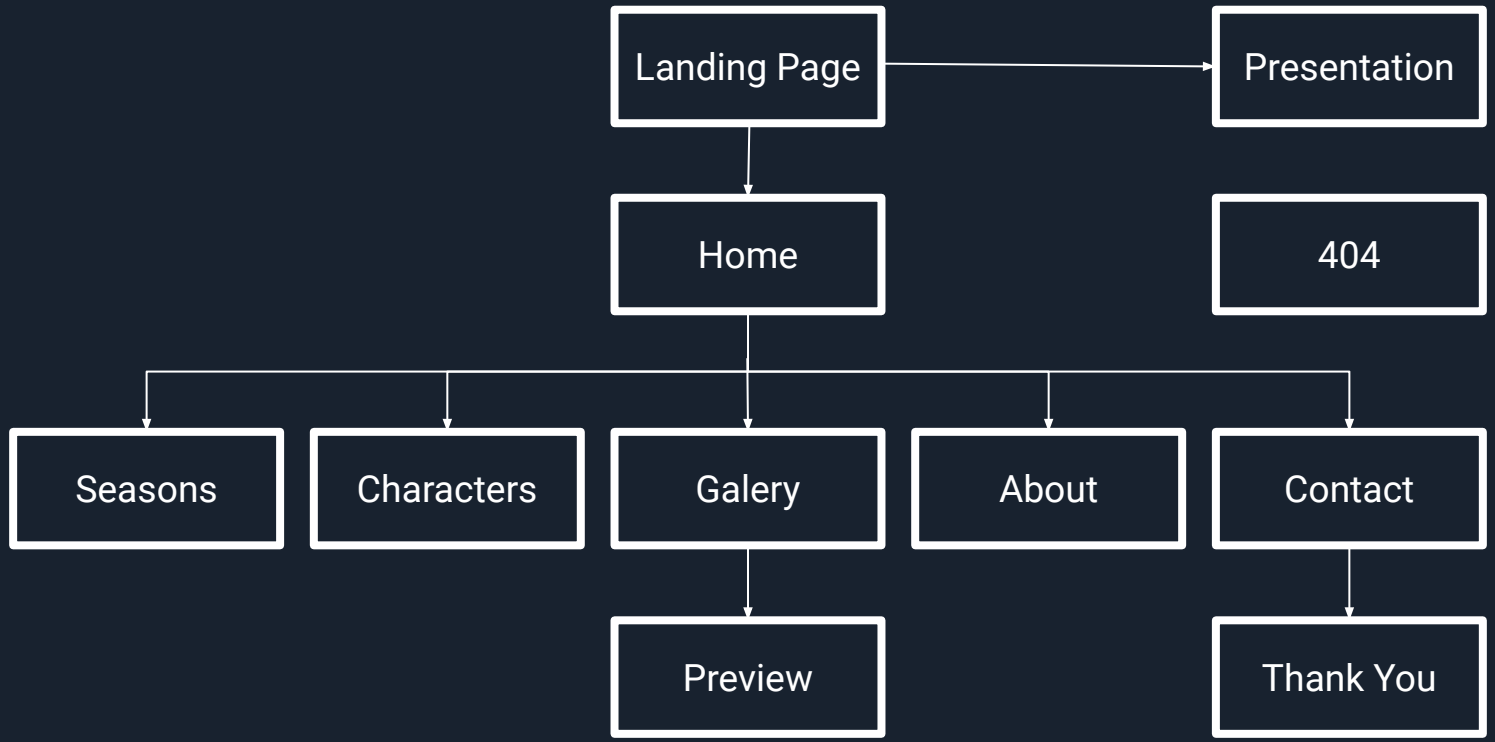
O projekte

Cieľom projektu bolo nadizajnovanie a naprogramovanie fanúšikovskej web stránky, zameranej na americký sci-fi seriál Stargate SG-1.

Túto tému pre vytvorenie web stránky som si vybral preto, lebo Stargate SG-1 je môj obľúbený seriál a už existujúce stránky, ktoré som našiel mi prídu zastaralé.

Na naprogramovanie web stránky som si vybral programovacie jazyky Python a JavaScript.
Pre backend som využil modul Flask. Na frontend som využil moje skúsenosti s HTML a CSS.

Návrh stránky



Hlavné komponenty stránky

Všetky hlavné komponenty, ktoré sa opakujú na skoro všetkých podstránkach ako Title, Favicon, Menu (navbar), Header, Footer, To-top button sú vytvorené iba raz a sú pridávané do placeholderov pomocou JavaScriptu.

```
<body>
    <div class="title-placeholder"></div>
    <div class="nav-placeholder"></div>
```

```
    <div class="totop-placeholder"></div>
    <div class="footer-placeholder"></div>
    <script src="../static/js/autoElement.js"></script>
    <script>StargateAutoElement();</script>
</body>
```

Takýmto spôsobom je možné robiť zmeny týchto hlavných komponentov na jednom mieste.

Navbar

Pri navrhovaní navbar-u som musel riešiť, aby zostal na vrchu viewport-u aj pri scrollovaní, aby bol vždy na celú šírku viewport-u a taktiež zvýrazňovanie vybranej podstránky.

Navbar som vytvoril pomocou button-ov, ktoré som následne naštýloval pomocou CSS. Zvýrazňovanie vybranej podstránky je riešené zmenou farby border-bottom.

```css
.stargate-nav nav button:hover {
    border-bottom: 4px solid var(--stargate-nav-line-color);
    transition: all 0.3s ease;
}

.stargate-nav nav .nav-button.current {
    border-bottom: 4px solid var(--stargate-nav-line-color);
}
```

Ostatné ovládacie prvky

V rámci projektu som nastavoval custom scroll bar cez CSS, aby ladil s dizajnom stránky.

Na ostatné ovládacie prvky ako To-top button, Scroll-down a ovládacie prvky náhľadu galérie som použil font-awesome (https://fontawesome.com)


```css
/* Custom ScrollBar */
/* width */
::-webkit-scrollbar {
    width: 10px;
}

/* Track */
::-webkit-scrollbar-track {
    background: #1E2529;
}

/* Handle */
::-webkit-scrollbar-thumb {
    background: #cbd4dd;
}

/* Handle on hover */
::-webkit-scrollbar-thumb:hover {
    background: #b8babb;
}

::-webkit-scrollbar-thumb:active {
    background: #D4D8DC;
}
/* ###### */
```

Home

Na úvodnej home stránke som chcel mať privítanie s animovanou šípkou na prescrollovanie k úvodným textom. Animáciu som riešil cez keyframes v CSS.

Pomocou nich je možné nastaviť meniace hodnoty objektu podľa časového úseku a vytvoriť tak animáciu.

Prescrollovanie k obsahu som vyriešil pomocou JavaScriptu so vstavanou funkciou window.scroll(), ktorá umožňuje posunutie na konkrétnu pozíciu.

```css
@keyframes bounce {
    0%, 20%, 50%, 80%, 100% {
        transform: translateY(0);
    }
    40% {
        transform: translateY(-30px);
    }
    60% {
        transform: translateY(-15px);
    }
}

.bounce {
    -moz-animation: bounce 2s infinite;
    -webkit-animation: bounce 2s infinite;
    animation: bounce 2s infinite;
}
```

```javascript
function scrollToElement(elementTag) {
    // Scrolls the page to the top of given element
    const element = document.querySelector(elementTag)
    let rect = element.getBoundingClientRect()
    window.scroll({
        top: rect.top+ window.scrollY - 65,
        left: 0,
        behavior: 'smooth'
    });
}
```

Seasons

Podmenu pre výber konkrétnej sezóny som riešil samostatnými panelmi s obrázkami jednotlivých sezón, ktoré sa rozbalujú na kliknutie.

Zbaľovanie a rozbaľovanie je riešené dynamickou zmenou šírky panelov pomocou JavaScript-u.


```javascript
const panels = document.querySelectorAll('.panel');
const active = document.querySelectorAll('.panel.active')

function calculateFlex(panel) {

    if ((2- (window.innerHeight / 977)) > 1) {
        let new_flex = (2- (window.innerHeight / 977)) * 14
        panel.style.flex = `${new_flex}%`
    }else if(2- (window.innerHeight / 977) === 1) {
        panel.style.flex = `10%`
    } else if (2- (window.innerHeight / 977) < 1) {
        panel.style.flex = `${(2- (window.innerHeight / 977)) * 18}%`
    }
}

function removeActiveClass() {
    panels.forEach((panel) => {
        panel.classList.remove('active')
        calculateFlex(panel)
})}

function panelOnClick (i, panel) {
    removeActiveClass();
    returnSeason(i);
    panel.classList.add('active');
    panel.style.flex = `40%`
    checkFooter();
}

panels.forEach((panel, i) => {
    panel.addEventListener('click', () => {
        // let new_flex = (2- (window.innerHeight / 977)) * 40
        panelOnClick(i+1, panel)
    })
})

window.addEventListener('resize',() => {
    panels.forEach((panel) => {
        if ((panel.classList.contains('active')) != true) {
            calculateFlex(panel)
        }
    })
});
```

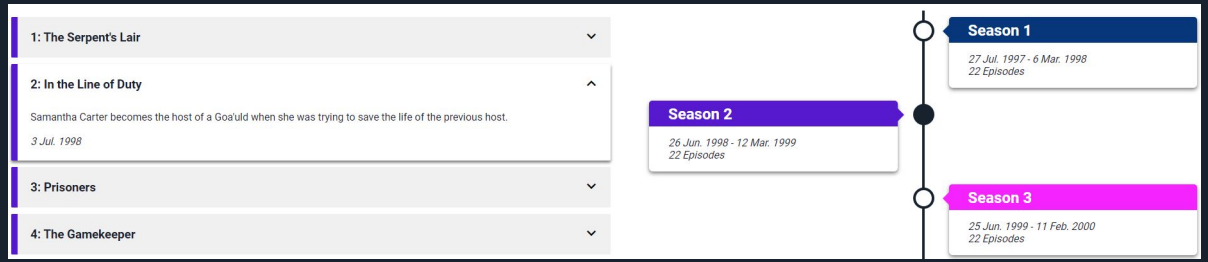
Seasons

V detaile zvolenej sezóny sa generuje cez for loop prehľad jednotlivých epizód a vygenerovanie časovej osi všetkých sezón.

V rámci generovania epizód sa mení automaticky farba podľa zvolenej sezóny. Informácie k jednotlivým epizódam je možné zbaliť a rozbaliť.

```
function returnSeason(seasonNum) {
    try {
        const episodesDiv = document.querySelector('.episodes')
        episodesDiv.innerHTML = `<h1 class="season-text">Season ${seasonNum}<br>
        <br>
        </h1>
        <div class="list-row">
            <div class="column left"></div>
            <div class="column right"><div class="timeline"></div></div>
        </div>`
        for (const [key, value] of Object.entries(seasons[seasonNum])) {
        if (value != null) {
            const leftColumn = document.querySelector('.column.left')
            leftColumn.innerHTML += createEpisodeElement(key, seasonNum)
        }

    }
    document.querySelectorAll('.card').forEach(toggle => {
        toggle.addEventListener('click', () => {
            toggle.classList.toggle('active')
        })
    })
    document.querySelectorAll('.card').forEach(card => {
        card.style.borderLeft = `10px solid ${colors[seasonNum]}`
    })
    document.querySelector('.column.right .timeline').innerHTML = createTimelineElement()

    document.querySelectorAll('.timeline-tab').forEach((tab, i) => {
        tab.style.setProperty('--season-color', colors[i+1])
        if (i+1 === seasonNum) {
            tab.style.setProperty('--dot-color', "var(--blue-gray-two)")
        }
    })

    } catch(err) {
        console.log(err.message)
    }
}
```

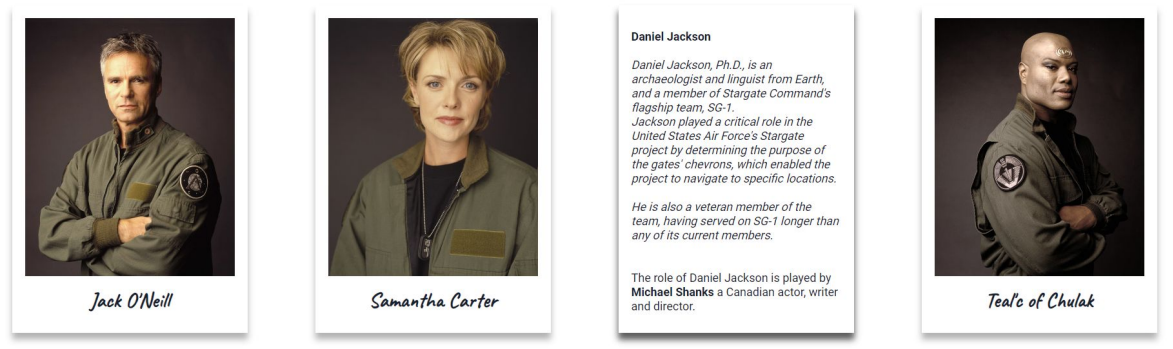
```
function createEpisodeElement(episodeNum,seasonNum) {
    let output = `<div class="episode" id="episode-${episodeNum}">
    <div class="card">
        <h3 class="card-title">${seasons[seasonNum][episodeNum]["title"]}</h3>
        <p class="card-text">${seasons[seasonNum][episodeNum]["description"]}<br><br>
        <i>${seasons[seasonNum][episodeNum]["date"]}</i></p>
        <div class="line"></div>

        <button class="card-toggle">
            <i class="fas fa-chevron-down"></i>
            <i class="fas fa-chevron-up"></i>
        </button>
    </div>
</div>`

    return output
}
```



Characters

Prehľad hlavných postáv som riešil ako mriežku s fotografiami. Pri presune kurzora nad fotografiu (hover) sa pomocou CSS pridá tieň a daná fotografia sa priblíži (scale).

Po kliknutí na fotografiu sa zobrazí popis postavy.




```
const characters = document.querySelectorAll('.character')

characters.forEach((character, index) => {
    character.addEventListener("click", () => {
        if (character.classList.contains("active")) {
            removeActive()
        } else {
            removeActive()
            character.classList.toggle('active')
        }
    })
    try {
        let card = document.createElement("div")
        card.classList.add('card')
        let text = document.createElement("p")
        text.innerHTML = charactersDescription[index+1]
        card.appendChild(text)
        character.appendChild(card)
    }catch {}
})
```

```
.character:hover .img-container img {
    transform: scale(1.05);
}

.character .card {
    -moz-user-select: none;
    -webkit-user-select: none;
    -ms-user-select:none;
    user-select:none;
    -o-user-select:none;
    visibility: hidden;
    opacity: 0;
    position: absolute;
    background-color: white;
    width: 100%;
    height: 100%;
    font-family: 'Roboto', sans-serif;
    padding: 20px;
    font-size: 18px;
    transition: visibility 0.5s, opacity 0.5s ease;
}
```

Galery

Po kliknutí na obrázok v galérií sa zobrazí jeho náhľad na celú obrazovku s nasledujúcimi možnosťami:

- Prepínanie medzi obrázkami klikaním na šípky na obrazovke, pomocou kláves A, D, šípka vľavo/vpravo, kolieskom myši
- Stiahnutie zvoleného obrázku na disk
- Uzatvorenie náhľadu kliknutím na ikonku s krížikom alebo stlačením ESC

```
const body = document.body
const slides = document.querySelectorAll('.slide')
const leftBtn = document.getElementById('left')
const rightBtn = document.getElementById('right')
const exitBtn = document.querySelector('.exit')
const downloadBtn = document.querySelector('.download')

let activeSlide = parseInt(document.getElementById("active-img").innerHTML, 10)-1

rightBtn.addEventListener('click', () => {
    activeSlide++

    if (activeSlide > slides.length - 1) {
        activeSlide = 0
    }

    setBgToBody()
    setActiveSlide()
    console.log(activeSlide)
})

leftBtn.addEventListener('click', () => {
    activeSlide--

    if (activeSlide < 0) {
        activeSlide = slides.length - 1
    }

    setBgToBody()
    setActiveSlide()
})

setBgToBody()
setActiveSlide()

function setBgToBody() {
    body.style.backgroundImage = slides[activeSlide].style.backgroundImage
}

function setActiveSlide() {
    let url = `../../static/images/stargate_galery_${activeSlide}.jpg`
    slides.forEach((slide) => slide.classList.remove('active'))
    slides[activeSlide].classList.add('active')
    if (activeSlide < 9) {
        url = `../../static/images/stargate_galery_0${activeSlide}.jpg`
    }
    downloadBtn.href = url
}
```

Galery

Pre stiahnutie obrázku sa dynamicky nastavuje url na zdrojový obrázok v adresári ./static/images

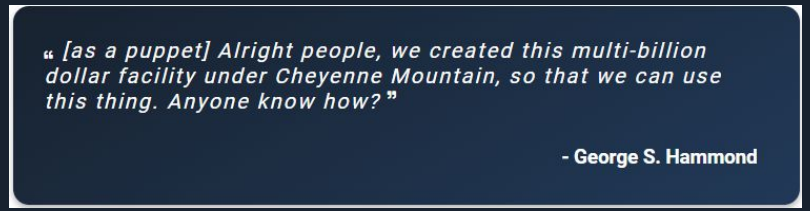
```
window.addEventListener('keydown',(event) => {
    if (event.code == "KeyA" || event.code == "ArrowLeft") {
        leftBtn.classList.add('active')
        activeSlide--
        if (activeSlide < 0) {
            activeSlide = slides.length - 1
        }
        setBgToBody()
        setActiveSlide()

    } else if (event.code == "KeyD" || event.code == "ArrowRight") {
        rightBtn.classList.add('active')
        activeSlide++
        if (activeSlide > slides.length - 1) {
            activeSlide = 0
        }
        setBgToBody()
        setActiveSlide()

    }else if (event.code == "Escape") {
        exitBtn.classList.add('active')}
})

window.addEventListener('keyup', (event) => {
    try {
        leftBtn.classList.remove('active')
    } catch {}
    try {
        rightBtn.classList.remove('active')
    } catch {}
    if (event.code == "Escape") {
        exitBtn.classList.remove('active')
        window.location.href='../galery'}
})

window.addEventListener('wheel', (event) => {
    if (event["deltaY"] > 0) {
        activeSlide++
        if (activeSlide > slides.length - 1) {
            activeSlide = 0
        }
        setBgToBody()
        setActiveSlide()
    } else if (event["deltaY"] < 0) {
        activeSlide--
        if (activeSlide < 0) {
            activeSlide = slides.length - 1
        }
        setBgToBody()
        setActiveSlide()
    }
})
```

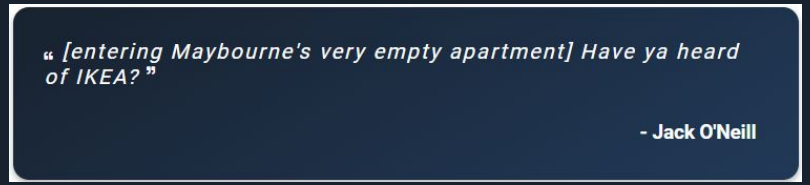
About

Je to statická podstránka s automaticky meniacimi sa citátmi zo seriálu. Citáty sú uložené v JavaScript liste.

Pri načítaní stránky sa vyberie náhodný citát a potom od vybraného citátu sa pokračuje v liste.

```
function randomIndex(choices) {
    var index = Math.floor(Math.random() * choices.length);
    return choices[index];
}

function generateQuoteEl() {
    let quote = randomIndex(quotes)
    let el = `<p class="quote-text"><i class="fas fa-quote-left"></i> ${quote["quote"]} <i class="fas fa-quote-right"></i></p>
<div class="quote-author"><h3>- ${quote["author"]}</h3></div>
`
    quotesEl.innerHTML = el
    return quotes.indexOf(quote)
}

let index = generateQuoteEl()

function nextQuoteEl() {
    let quote = quotes[index]
    if(index>quotes.length-1){
        index = 0
    } else {index++}

    let el = `<p class="quote-text"><i class="fas fa-quote-left"></i> ${quote["quote"]} <i class="fas fa-quote-right"></i></p>
    <div class="quote-author"><h3>- ${quote["author"]}</h3></div>
    `
    quotesEl.innerHTML = el
}

setInterval(nextQuoteEl, 10000)
```





Contact

Podstránka s kontaktným formulárom, prepojeným na gmail konto.

V rámci formulára sa pri každom načítaní stránky vyberú náhodné placeholdre pre textové polia a text tlačítka na odoslanie.

Odoslanie samotnej správy z formulára je riešené pomocou Python modulu SMTPLib.

Po odoslaní správy sa zobrazí Thank you page, ktorá je nastavená na zobrazenie pri použití POST http method na contact podstránke.

```python
import smtplib, ssl
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart


def sendMail(sender, name,subject, messageContent):
    smtp_server = "smtp.gmail.com"
    sender_email = "stargate.sg1.fanpage@gmail.com"
    receiver_email = "stargate.sg1.fanpage@gmail.com"
    password = '              '

    message = MIMEMultipart()
    message["Subject"] = f"{subject}"
    message["From"] = sender
    message["To"] = receiver_email

    html = f"""\
    <html>
        <body>
        <p><b>Name:</b> {name}<br><b>Email:</b> {sender}<br></p>
        <p><b>Message:</b> <br>{ messageContent }</p>
        </body>
    </html>
    """

    # Turn these into plain/html MIMEText objects
    part = MIMEText(html, "html")

    # Add HTML/plain-text parts to MIMEMultipart message
    # The email client will try to render the last part first
    message.attach(part)

    # Create secure connection with server and send email
    context = ssl.create_default_context()
    with smtplib.SMTP_SSL(smtp_server, 465, context=context) as server:
        server.login(sender_email, password)
        server.sendmail(
            sender_email, receiver_email, message.as_string()
        )
```

Error handling

V rámci backend-u Python Flask je možné nastaviť vlastné error podstránky.

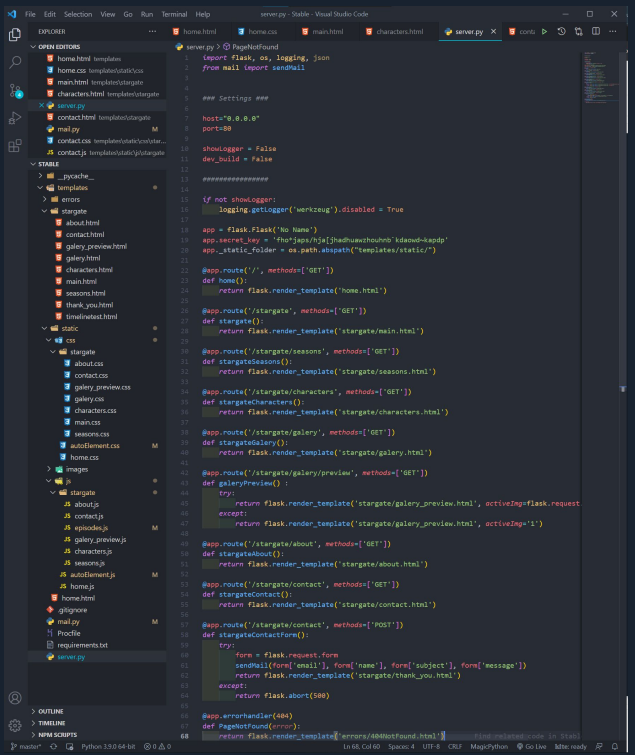
V projekte je takto ošetrená chyba 404 teda “Page not found”, ale obdobne je možné nastaviť aj vlastné error podstránky aj pre ďalšie chyby, ako napríklad 4xx “Client error”, 5xx “Server Error”

```python
@app.errorhandler(404)
def PageNotFound(error):
    return flask.render_template('errors/404NotFound.html')
```

Development

V rámci projektu som používal následujúce nástroje

- Visual Studio Code (vývojové prostredie)
- Kite (automatické doplňovanie príkazov)
- GitHub (verzionovanie kódu)
- Adobe Photoshop (úprava obrázkov)
- MS PowerToys Preview (color picker)



Debugging

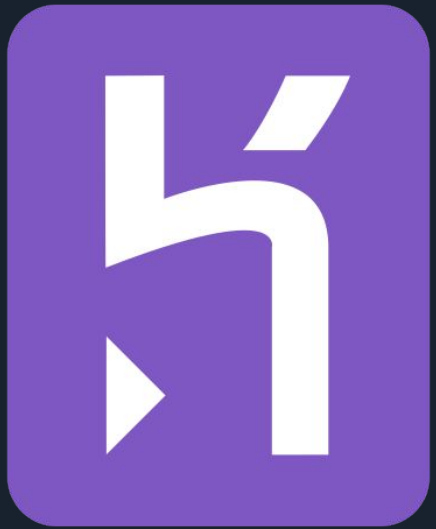
Na debugovanie kódu som používal Chrome Dev Tools, ktoré sú dostupné cez F12.

Tento nástroj som používal hlavne na kontrolovanie JavaScriptovej konzoly a na otestovanie si menších zmien priamo v HTML, aby som videl rozdiel bez úpravy zdrojového kódu.

Hosting

Hosting Heroku som si vybral preto, že podporuje Python, je bezplatný a bol celkom jednoduchý na nastavenie.

Heroku používa na konfiguráciu súbor Procfile, v ktorom som musel definovať worker web, aby ich firewall sprístupnil moju web stránku na internete. Moduly potrebné pre backend sa definujú v súbore requirements.txt.



Procfile

```
1   web: gunicorn server:app
```

requirements.txt

```
1   flask>=1.1.2
2   gunicorn>=20.0.4
```

Vylepšenia do budúcna

- Responzivita pre rôzne zariadenia a prehliadače
- Lepšia kompatibilita CSS s rôznymi prehliadačmi
- Samostatná verzia navbar-u pre mobilné zariadenia
- Zjednodušenie a upratanie CSS
- Doplnenie reCaptcha do kontaktného formulára, alebo inej ochrany pred botmi
- Pridať ďalší obsah (viac textov, blog, kvíz)
- Zakúpenie vlastnej domény a prenesenie na výkonnejší platený hosting

Version 2.0

Záver

Stránka je dostupná na url:
https://stargate-fanpage.herokuapp.com

Zdrojový kód dostupný na GitHub:
https://github.com/mindstormjak/Stargate-Fanpage_Competition

Na webe boli použité obrázky vlastnené spoločnosťou MGM.
Niektoré časti textov pre vytvorenie obsahu pochádzajú
z imdb.com, wikipedia.org a fandom.com

Jakub Krčmárik
jakubko.krcmarik@gmail.com