Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Ročníkový projekt

Autor: Filip Novák
Odbor: Informatika
Ročník: druhý
Školský rok: 2018/2019

Práca v zimnom semestri

Vytvoril som localhost server v c++ na porte 8080, ktorý prijíma HTTP požiadavky od HTTP klienta (webový prehliadač) a posiela naspäť HTTP odpoveď vo formáte, ktorý webový prehliadač od nás očakáva.

Práca v letnom semestri

Úlohou bolo dokončiť back-end (server) webovej aplikácie a spojiť ho s front-endom, aby aplikácia fungovala a dala sa používať.
Od minulého semestra som zmenil parsovaciu funkciu na takú, ktorá si z požiadavky, ktorú mi poslal klient zapamätala a vybrala len HTTP metódu, PATH z URL a DATA. Pridal som rozdelenia podľa toho aká HTTP metóda alebo PATH mi príde, aby som vedel ako mám spracovať požiadavku. Následne som ešte v tomto semestri spravil classu GraphBuilder, ktorá slúži na to, aby si vytvorila graph, ktorý si klient vo svojom webovom prehliadači nakreslil. Nakoniec som vytvoril struct Response a k nej funkciu make_response, ktorá vytvára HTTP odpoveď, ktorú posielame klientovi.

Obsah a funkčnosť súborov v ročníkovom projekte

V **main.cpp** je kód servera a teda je to zdrojový súbor, na ktorom to všetko beží. Klient odošle požiadavku, ktorá príde na server a uloží sa do **buffera**. Potom si vytvoríme premennú typu Request s menom **request** a zavoláme metódu **parse** na premennú **parser** typu HttpRequestParser, ktorej kód je napísaný v súbore **httprequestparser.h**. Metóda **parse** nám umožní použiť **request.method**, ktorá nám vráti HTTP metódu (napr. GET, POST), **request.uri** nám vráti PATH a do stringu **data** uloží DATA. Následne sa pýtame či **request.method** je POST alebo GET.

GET požiadavka:
Pýtame sa či nám PATH prišla v tvare **/multipol** alebo **/invariants**. PATH **/multipol** nám prichádza vtedy, keď klient spustí prehliadač s webovou aplikáciou. V tomto prípade sa nám snaží povedať, aby sme mu poslali rôzne typy multipólov, s ktorými potom môže kresliť. Aby sme splnili túto požiadavku, tak si zavoláme funkciu **datab_mult**, ktorá je v main.cpp. Funkcia si skopíruje obsah súboru **array_of_multipoles.txt**, v ktorom sú informácie o rôznych multipóloch. Táto funkcia nám vráti json, ktorý premeníme na string a pošleme ho ako paramater do funkcie **make_response** v main.cpp, ktorá s použitím structu **Response** vytvoria HTTP odpoveď, ktorá sa potom môže poslať klientovi.
Ak nám príde PATH v tvare **/invariants** to znamená, že klient stlačil tlačidlo **INVARIANTS** v aplikácii a teda poslal kód grafu v kódovaní sparse6, ktorý sme už predtým vyrobili a chce od nás rôzne informácie o tomto grafe. S použitím funkcií z knižnice **ba_graph** v súbore **relatko-ba-graph-52fc092b557e** vypočítame rôzne invarianty poslaného grafu, ktoré si ukladáme do json formátu. Následne znova zavoláme funkciu **make_response**, do ktorej dáme ako jeden z parametrov json s invariantami a potom pošleme odpoveď.

POST požiadavka:

Pri tejto požiadavke je len jedna možnosť, ktorá je akceptovaná a to PATH v tvare **/graph**. Prichádza vtedy, keď klient stlačí tlačidlo **SEND GRAPH** v aplikácii. Klient nám posiela v DATA informácie ohľadom grafu, ktorý si vo webovej aplikácii nakreslil. Sú to informácie, z ktorých si ten graf v našom servery vytvoríme pomocou funkcií z knižnice **ba_graf** v súbore **relatko-ba-graph-52fc092b557e**. Vytvoríme si inštanciu classy **GraphBuilder**, na ktorej zavoláme metódu **make_graph**. V metóde **make_graph** si pomocou informácií z DATA a informácií z textového súboru **array_of_multipoles.txt** vytvoríme postupne graf, ktorý na konci pomocou funkcie z **ba_graph** zakódujeme do sparse6. Tento kód grafu pošleme ako parameter funkcii **make_response**, ktorá vytvorí odpoveď, v ktorej bude ten kód grafu. Správu potom pošleme klientovi.

Ak by sa počas behu servera stal neočakávaný problém, napríklad ak by sa stala chyba pri parsovaní požiadavky alebo by nám došla iná metóda než POST a GET, alebo príde PATH, ktorú nevieme spracovať, tak zavoláme **make_response**, ktorá nám vyhotoví error správu. Správa sa pošle klientovi, ktorého upozorní, že sa stala nejaká chyba.
Na serveri ako output do terminálu vypisujeme vždy prichádzajúcu požiadavku a aj odpoveď, ktorú posielame klientovi.

Kompilácia a spustenie servera

Požiadavka na kompiláciu servera je taká, že treba mať g++ kompilátor verzie aspoň 8!!!

- treba si stiahnuť súbory z https://github.com/RadoslavSmarzik/graph-editor
- otvoriť súbor **Server**
- otvoriť **Makefile**
- v Makefile si treba upraviť cestu ku súboru **relatko-ba-graph-52fc092b557e** v CFLAGS.
- cez príkazový riadok sa treba dostať do súboru **Server**
- napísať príkaz **make**, ak prebehla kompilácia úspešne, potom:
    - ak používate Windows, stačí napísať **server**
    - ak používate Linux, stačí napísať **./server**

  Server začal počúvať a čaká na požiadavky.