Tehtävän kuvaus ja työn rajaus

Tehtävänanto oli smartpost-sovellus case, jossa oli määritelty ohjelmiston toiminnallisuudet ja ominaisuudet.

Tietokannan käsitemalli ja eheyssäännöt

kuvattu erikseen tietokantaraportissa.

Ohjelman kuvaus

Ohjelma toteuttaa harjoitustyön tehtävänannon mukaisen toiminnan sekä laajennettuja ominaisuuksia logien ja käyttäjätilien muodossa.

Ohjelman toteutus

Ohjelmassa on postipiste-olioita, jotka luetaan tietokantaan xml-lukijaoliolla ja siitä haun mukaisesti javascript-rajapinnalta kartalle graafiseen käyttöliittymään. Pisteiden välille voidaan luoda polkuja, mikäli lähetys-oliolla on vaadittavat ominaisuudet. Lähetysoliolla on sisällään lähtö-, ja päätepiste sekä lähetettävä paketti-olio sekä tietokantatunnisteet.

Paketissa on tunnisteiden lisäksi pakettiluokka sekä esine. Esineen mahtuminen pakettiin tarkistetaan sitä luodessa. Paketti-tietokanta toimi tehtävänannon mukaisena varastona.

Ohjelmassa on mahdollista luoda uusia postipisteitä ja esineitä tietokantaan, jossa ne toimivat kuten jo valmiiksi syötetyt esineet ja pisteet. Nämä lisäykset tapahtuvat omissa ikkunoissaan.

Lähetys saattaa hajota, jos lähetyksen särkyvyys tai kokoparametrit eivät läpäise timotei-olion tarkistusta. Tämä olio sisältää satunnaislukugeneraattorin, joka mallintaa ”todellista työmiestä” ja tämän huonoja ja hyviä päiviä.

Kaikki lähetetyt paketit tallennetaan logiin sekä tietokantaan että ulkoiseen log.txt-tiedostoon, jos käyttäjä näin tahtoo. Tähän käytetään IOfileoperator-oliota.

Ylimääräisenä ominaisuutena ohjelmassa on käyttäjätilit, joita voi luoda ja jotka säilyvät tietokannassa. Lähetyksiä ja logeja voi tarkastella vain oman käyttäjätilin osalta activeUser-olion ja tietokannan kayttajat-taulun active-boolean arvoa hyödyntäen.

Tietokantaskeeman ja alkuperäiset insertit luettiin javan kautta tietokantaan. Kyseinen koodinpätkä on kommentoitu pois alkuikkunan koodista, jottei tietokanta loisi uudestaan itseään joka käynnistyskerralla (ja kaatuisi).

luokkakaavio

kts. repositorio

yhteenveto

Mitä tästä opittiin; aloita työt ajoissa, älä koodaa liian monta tuntia putkeen. Työteho heikkenee ja koodista tulee rumaa. Piti tehdä uusiksi monta asiaa, jotka olivat yksinkertaisia nukkuneilla aivoilla. SQLiten virhetiedot ovat välillä vähän kryptisiä, mutta rubberduck-debuggaamalla saatiin todella hyviä tuloksia työtä tehdessä.

Mitä paremmin osa oli suunniteltu, sen paremmin se toimi. Esimerkiksi oivallus tietokannan käytöstä tiedon välittämisessä ikkunasta toiselle oli hyvin oleellinen läpimurto.