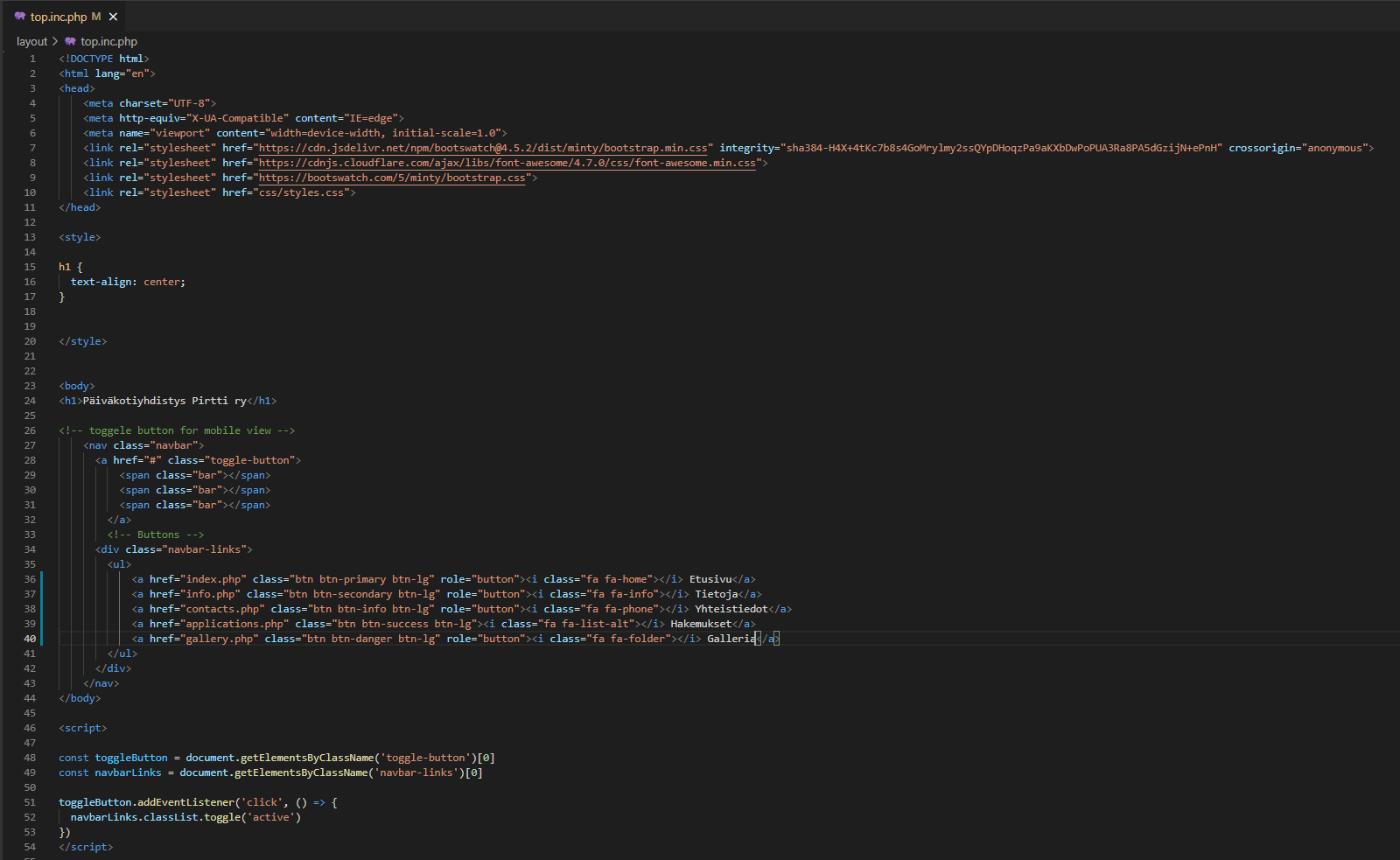
**Päiväkotiyhdistys ry nettisivujen koodi**

* Päiväkotiyhdistys ry nettisivun ulkonäköön on käytetty bootstrappia.
* Sivuun on käytetty myös seuraavia HTML, JavaScriptiä, CSS, PHP & SQL Tietokantaa.



top.inc.php



* Tässä on yläosa sivustosta, joka sisälletään jokaiselle skriptille nettisivulla. Tämä on perus HTML & lopussa vähän JS koodia. Lisään ekaksi kaikki tarvittavat bootsrapt scriptit ja CSS skripti.
* Laitan otsikko tekstin keskelle CSS:ssällä.
* Bodyssa on otsikko, sitten tein mobiiliin (hamppari painikkeen) joka avaa ja piilottaa painikkeet jotka on alhaalla listattu.
* Pääset painikkeilla etusivu-, tietoja-, yhteistiedot-, hakemus- & galleria sivustoille.
* Viimeseksi on js koodi, kun hamppari painike on aktiivinen painalluksesta se avaa painikkeet.

css/styles.css



* Säätelen containerille tyylittelyä.
* Laitan background kuvan sivustolle.
* Linkeille koot.
* Säätelen missä hamppari painike näkyy.
* Hamppari näppäimen koko.
* Media osuudessa millä kokolla hamppari painike tulee näkyviin.
* Piilotan linkit kun hamppari näppäin tulee näkyviin ja linkit tulee vaan näkyviin ku näppäin on aktiivinen.
* Määritän millä tavalla linkit tulee näkyviin.

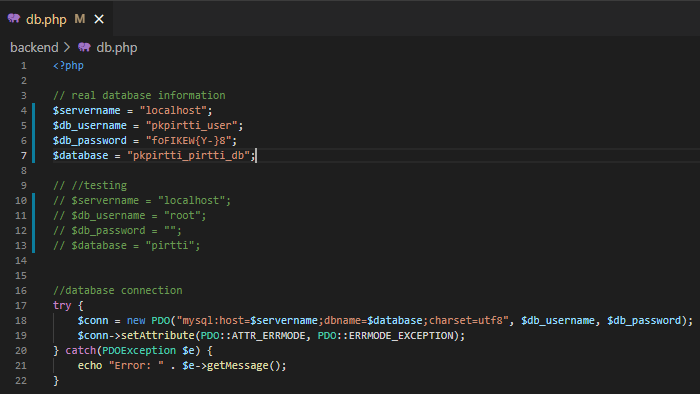
bot.inc.php

Text

Description automatically generated

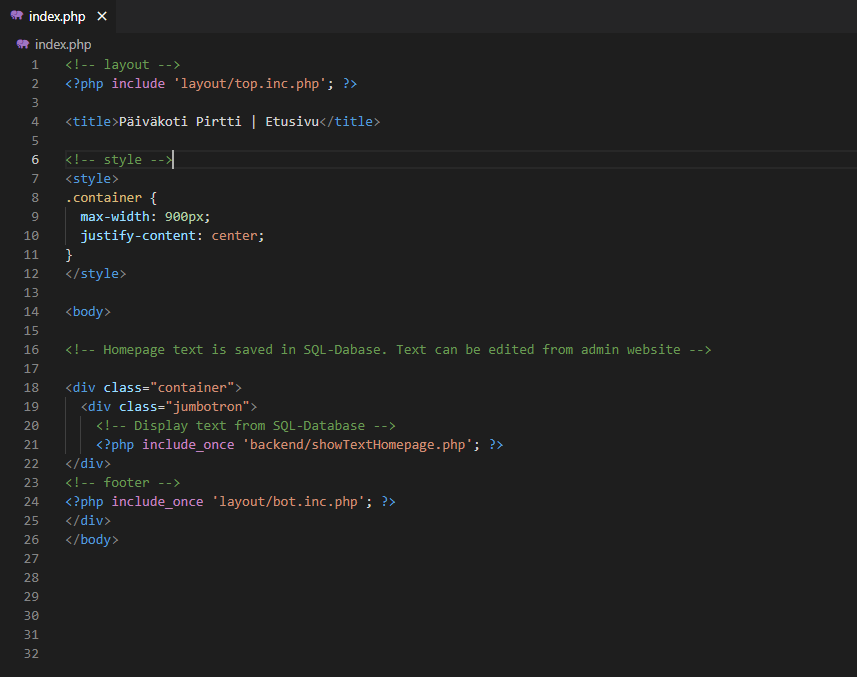
* Tässä on alapuoli, joka myös liitetään jokaiseen scriptiin. Alatunnisteessa lukee copyright & päiväkohdistus Pirtti ry.

db.php



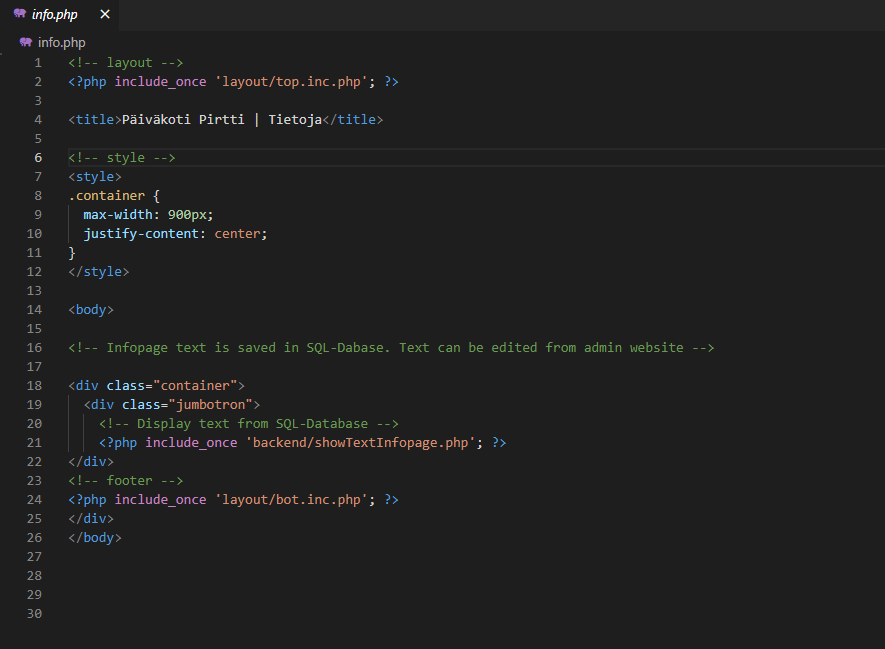
* Tietokantayhteys PDO:lla

index.php



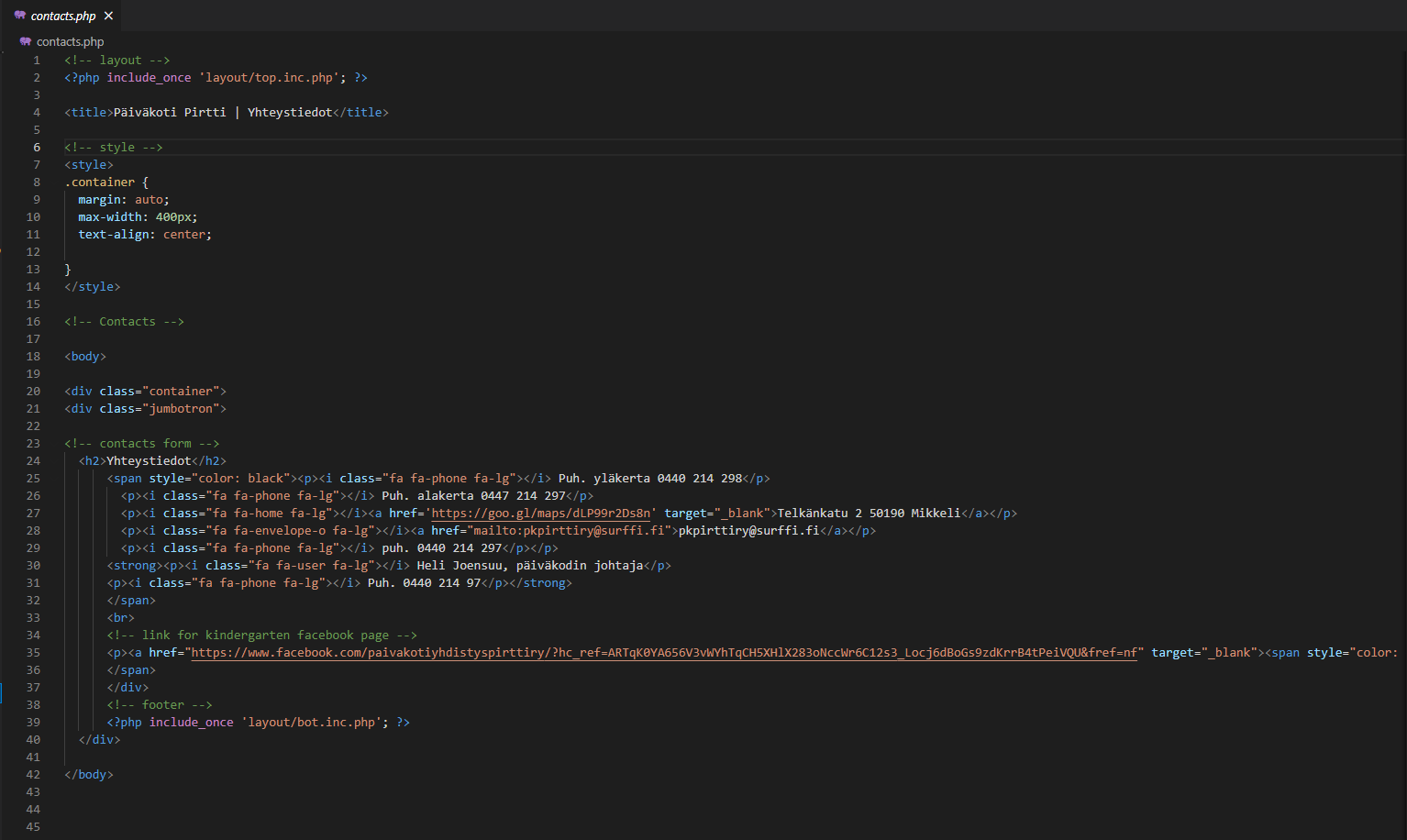
* Tässä on etusivu.
* liitetään yläosa.
* Muokataan koko containeriin.
* Haetaan tietokannasta teksti. Tiedonhallinta sivustolla voi muokata etusivun tekstiä tällä tavoin.
* Lisätään alaosa.

info.php



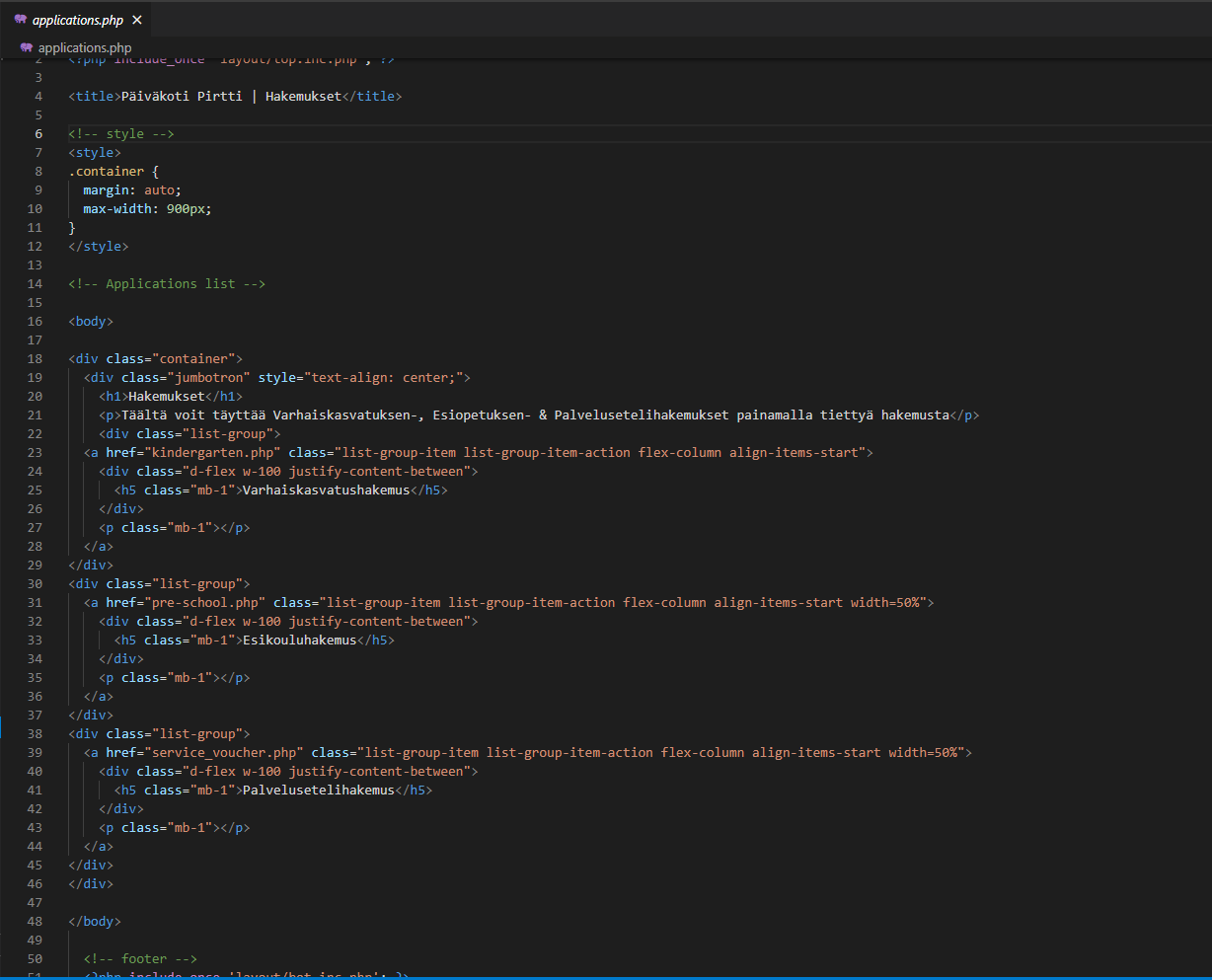
* Tässä on tietojasivu.
* liitetään yläosa.
* Muokataan koko containeriin.
* Haetaan tietokannasta teksti. Tiedonhallinta sivustolla voi muokata tietojasivun tekstiä tällä tavoin.
* Lisätään alaosa.

contacts.php



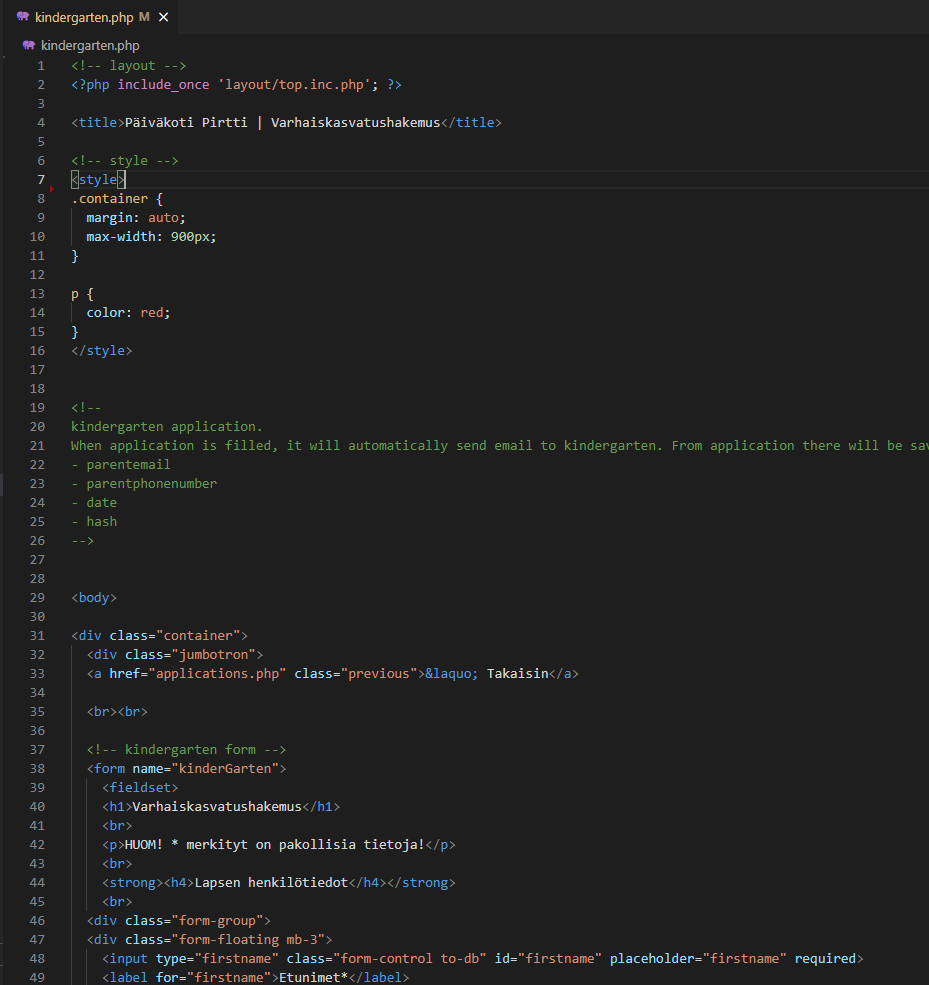
* Tässä on yhteistietosivu.
* liitetään yläosa.
* Muokataan koko containeriin.
* Formissa on päiväkodin yhteistiedot lueteltu. Span tyylillä laitan tekstin mustaksi ja classissä laitan tietyille riveille kännykkä-, koti-, maili-, käyttäjä ikonit.
* Lisäsin facebook logon lopuksi, jolla pääsee päiväkodin facebook sivustolle. Sivusto avautuu uudessa välilehdessä.
* Lisätään alaosa.

applications.php



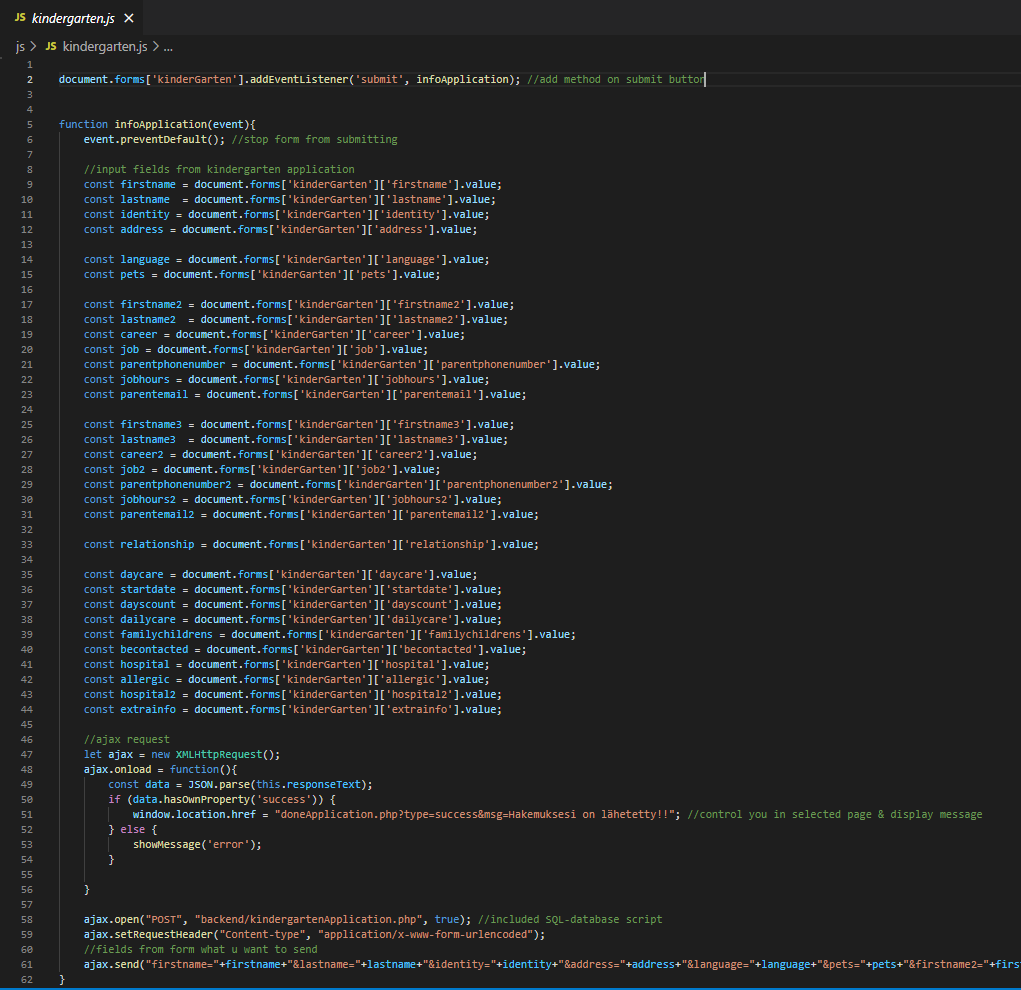
* Tässä on hakemussivusto.
* liitetään yläosa.
* Muokataan koko containeriin.
* Formiin lisätään Varhaiskasvatus-, Esikoulu- & Palvelusetelihakemus linkit jolla pääsee tietylle sivustolle.
* Lisätään alaosa.

kindergarten.php



* Tässä on Varhaiskasvatushakemus. Koko form osioo en laittanut tähän. Se on pitkä jotenka päätin laittaa alku ja loppuosan.
* liitetään yläosa.
* Muokataan koko containeriin. Lisätään <p> HUOM! osioon punainen teksti.
* Takaisin painike jolla pääsee hakemussivustolle takaisin.
* Form osio joka ei ole tässä kokonaan mutta siellä on perus lapsentietoja, huoltajienntietoja ja lapsesta vähä lisäkysymyksiä kysytty.
* Submit painike jolla lähetät hakemuksen. Painikkeeseen on tehty JS:ssällä tapahtuman käsittelijä.
* Liitetään JS skripti.
* Lisätään alaosa.

js/kindergarten.js

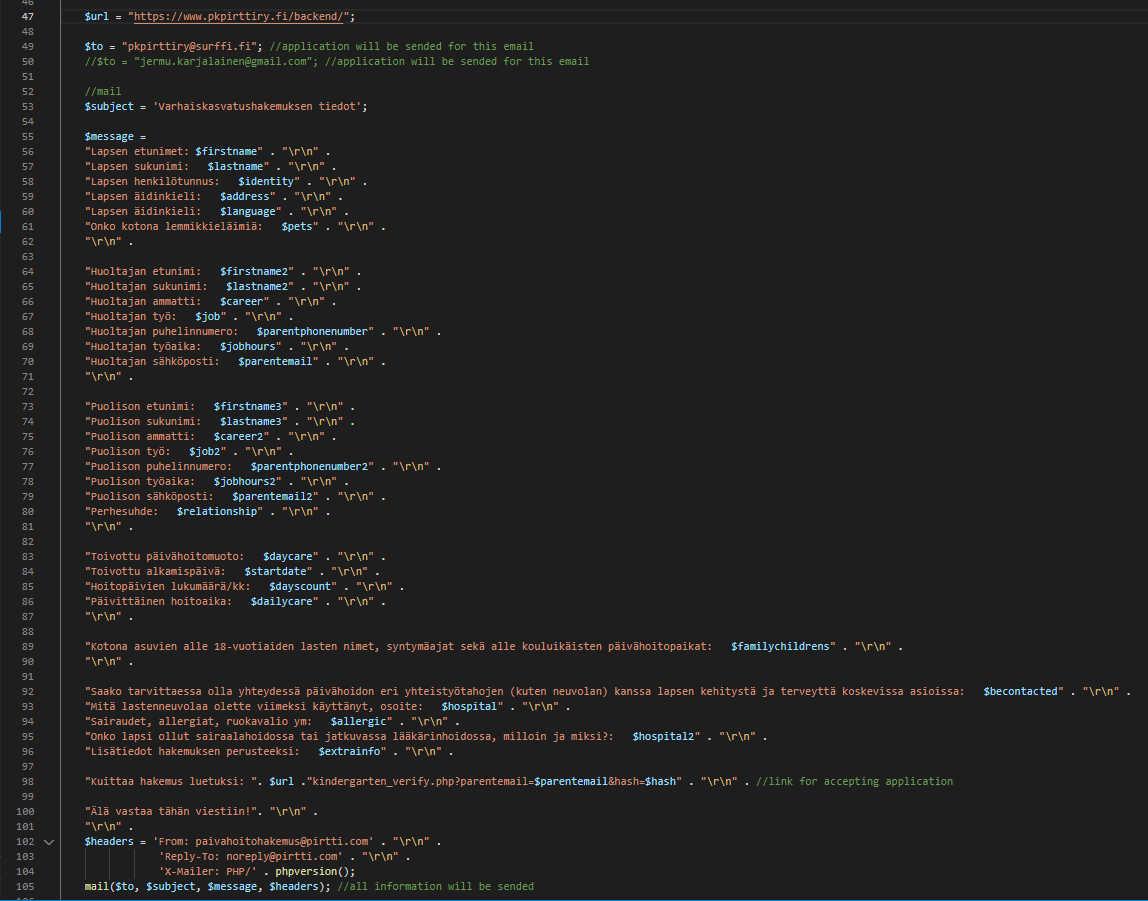


* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on koko formin kentät ja otetaan ne talteen.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

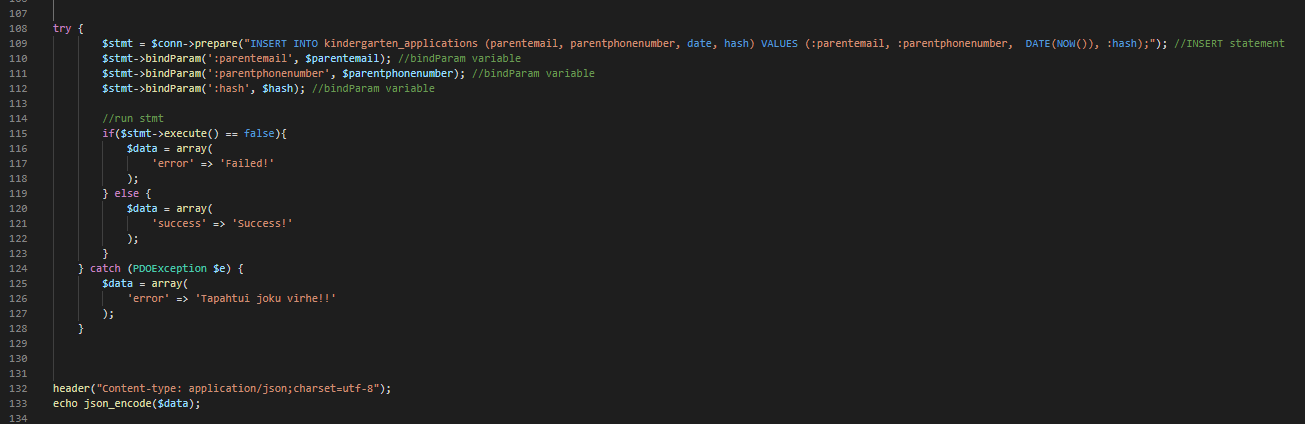
backend/kindergartenApplication.php



* Lisätään tietokanta yhteys.
* Otetaan POST-data vastaan tietokanta tallennusta varten.
* Muutetaan POST-data tiedot Hash:ksi.

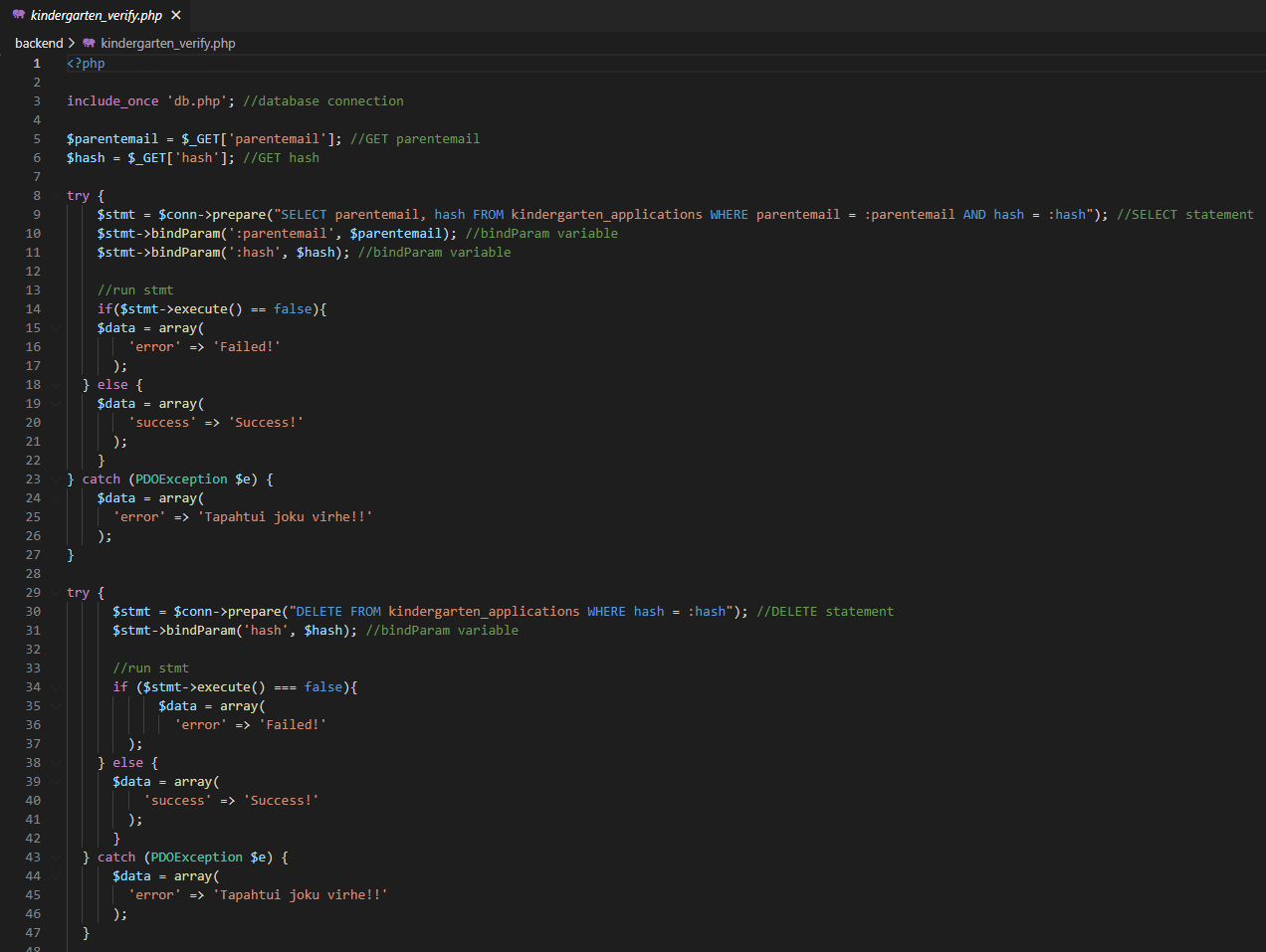


* Luodaan sähköposti kaikista tiedoista, jotka lähetetään päiväkotiin.
* ”kuittaa hakemus luetuksi” päästään uudelle skriptille jolla hyväksytään hakemus.
* mail($to, $message, $header);
* $subject = sähköpostin aihe
* $to = kenelle sähköposti lähetetään
* $message = sähköpostin viesti
* $header = keneltä sähköposti on tullut

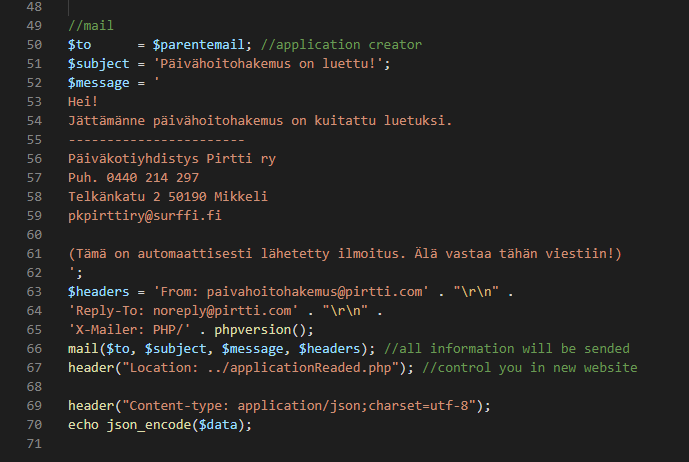


* Insert lausekkeella tallennetaan tietokantaan seuraavat tiedot parentemail, parentphonenumber, date & hash.
* Hash sisältää kaiken POST-datan.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

Backend/kindergarten\_verify.php

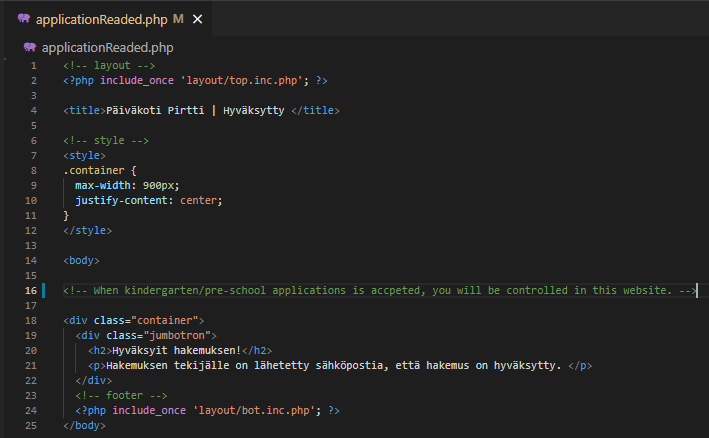


* Lisätään tietokanta yhteys.
* Otetaan tietokannasta GET-parameterillä parentemail & hash.
* Luodaan SELECT-lauseke jolla valitaan parentemail & hash rivit.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja. Jos onnistuu päästään DELETE-lausekkeeseen jolla poistetaan hashin avulla hakemus tietokannantaulusta.



* Kun hakemus on hyväksytty ja poistettu tietokannasta lähetetään viesti hakemuksen tekijälle ($parentemail), että hakemus on luettu.

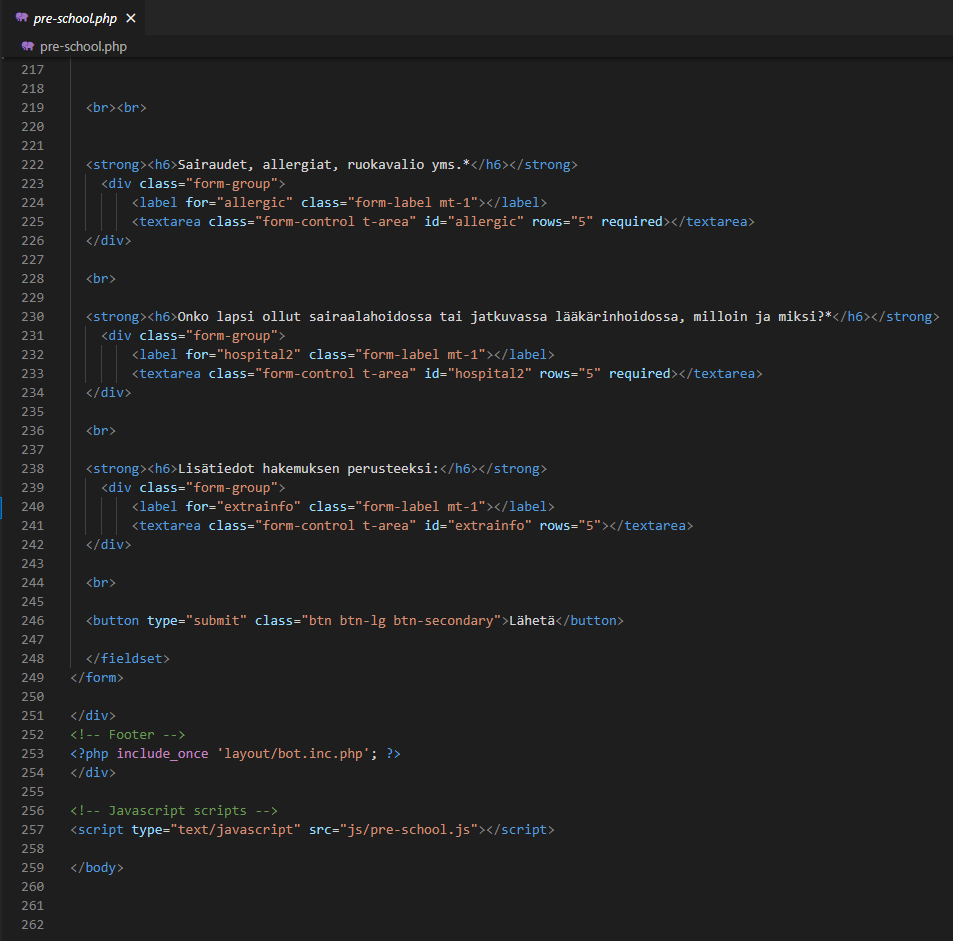
applicationReaded.php



* Hakemuksen kuittaaja pääsee sähköpostin linkkiä painamalla sivulle jossa on kerrotaan, että hakemuksen tekijälle on lähetetty sähköpostia hakemuksen hyväksynnästä.

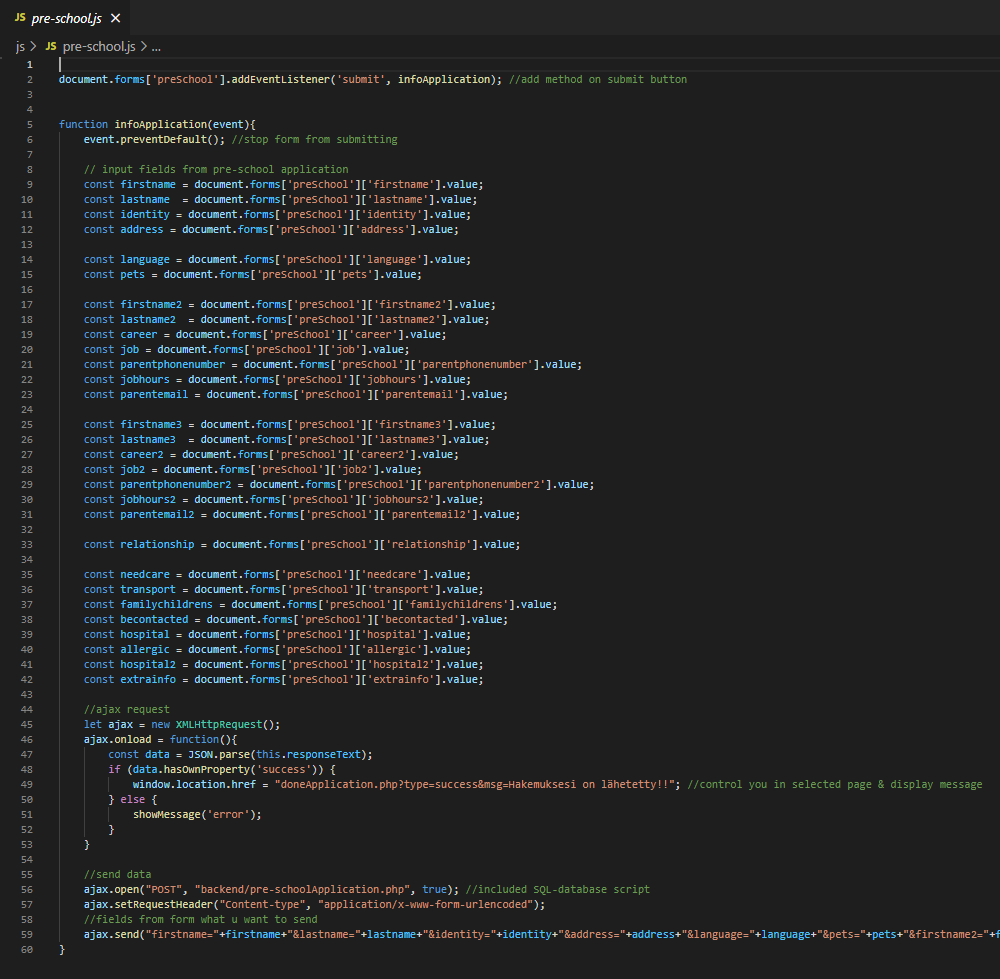
pre-school.php

Text

Description automatically generated

* Tässä on Esikouluhakemus. Koko form osioo en laittanut tähän. Se on pitkä jotenka päätin laittaa alku ja loppuosan.
* liitetään yläosa.
* Muokataan koko containeriin. Lisätään <p> HUOM! osioon punainen teksti.
* Takaisin painike jolla pääsee hakemussivustolle takaisin.
* Form osio joka ei ole tässä kokonaan mutta siellä on perus lapsentietoja, huoltajienntietoja ja lapsesta vähä lisäkysymyksiä kysytty.
* Submit painike jolla lähetät hakemuksen. Painikkeeseen on tehty JS:ssällä tapahtuman käsittelijä.
* Liitetään JS skripti.
* Lisätään alaosa.

js/pre-school.js

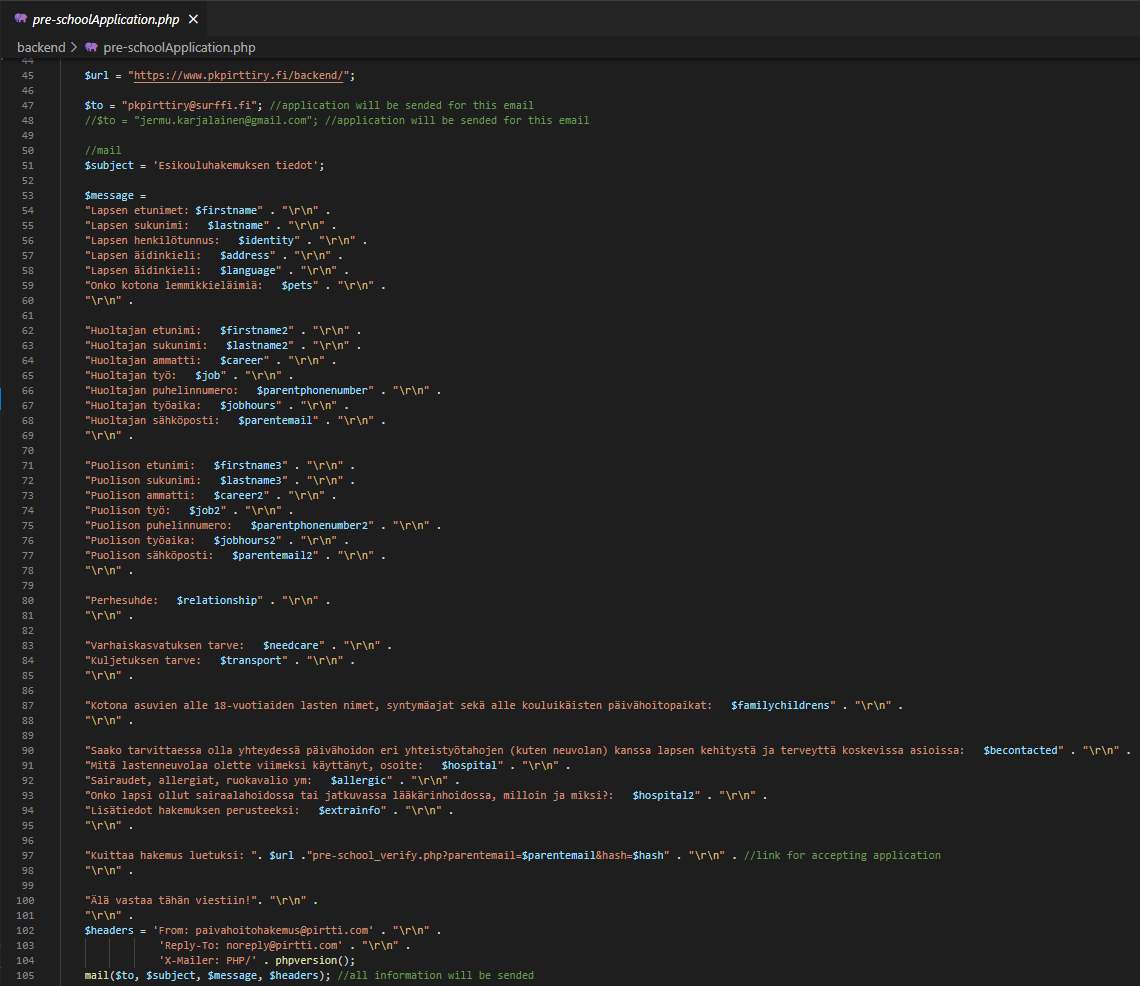


* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on koko formin kentät ja otetaan ne talteen.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

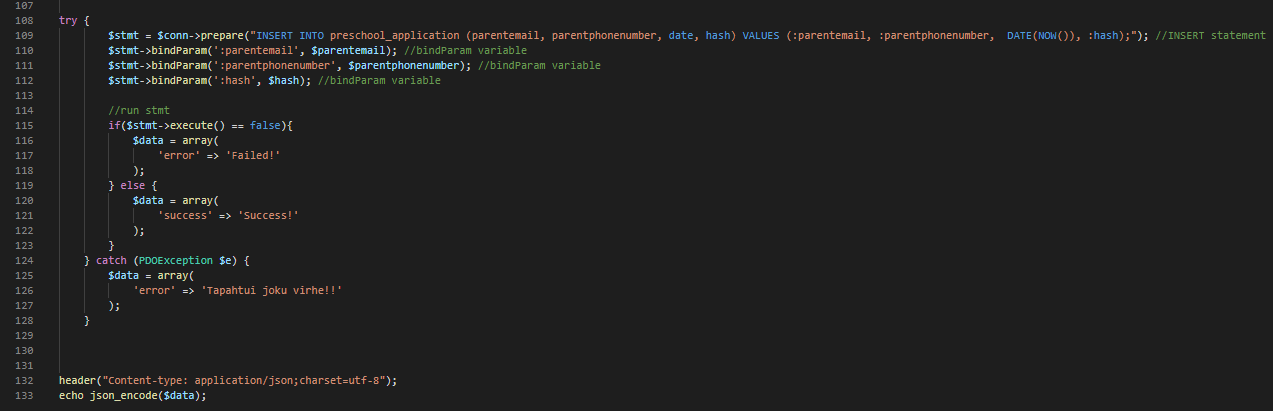
Backend/pre-schoolApplication.php



* Lisätään tietokanta yhteys.
* Otetaan POST-data vastaan tietokanta tallennusta varten.
* Muutetaan POST-data tiedot Hash:ksi.

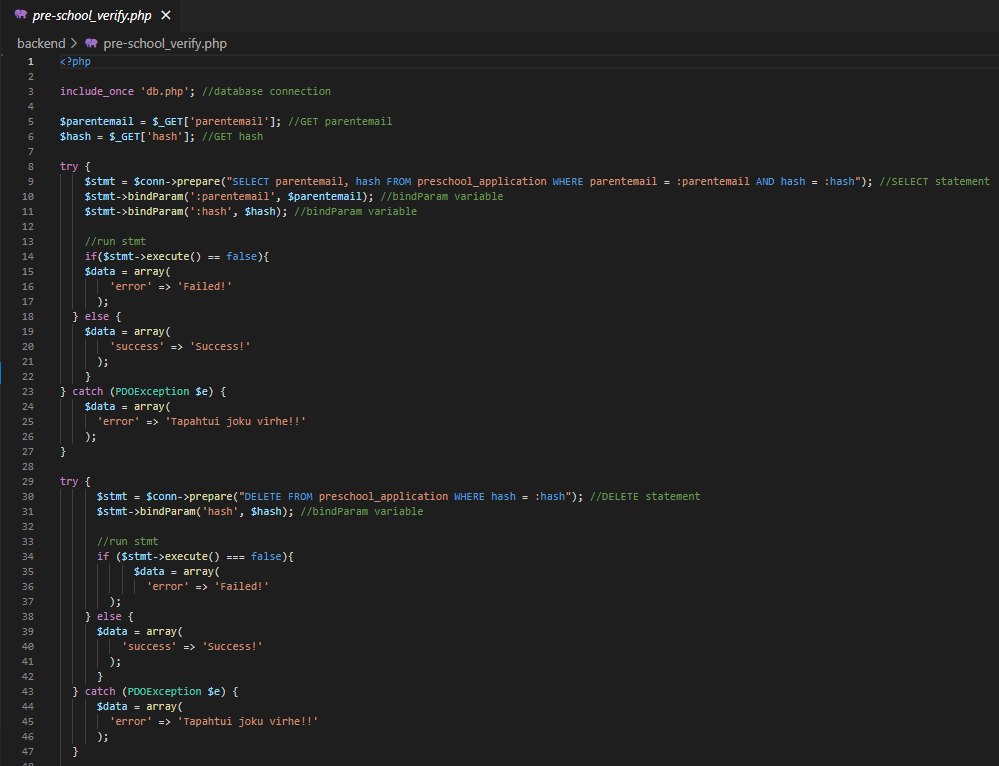


* Luodaan sähköposti kaikista tiedoista, jotka lähetetään päiväkotiin.
* ”kuittaa hakemus luetuksi” päästään uudelle skriptille jolla hyväksytään hakemus.
* mail($to, $message, $header);
* $subject = sähköpostin aihe
* $to = kenelle sähköposti lähetetään
* $message = sähköpostin viesti
* $header = keneltä sähköposti on tullut

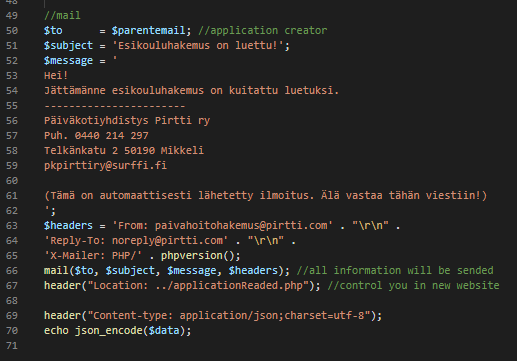


* Insert lausekkeella tallennetaan tietokantaan seuraavat tiedot parentemail, parentphonenumber, date & hash.
* Hash sisältää kaiken POST-datan.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

Backend/kindergarten\_verify.php



* Lisätään tietokanta yhteys.
* Otetaan tietokannasta GET-parameterillä parentemail & hash.
* Luodaan SELECT-lauseke jolla valitaan parentemail & hash rivit.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja. Jos onnistuu päästään DELETE-lausekkeeseen jolla poistetaan hashin avulla hakemus tietokannantaulusta.



* Kun hakemus on hyväksytty ja poistettu tietokannasta lähetetään viesti hakemuksen tekijälle ($parentemail), että hakemus on luettu.

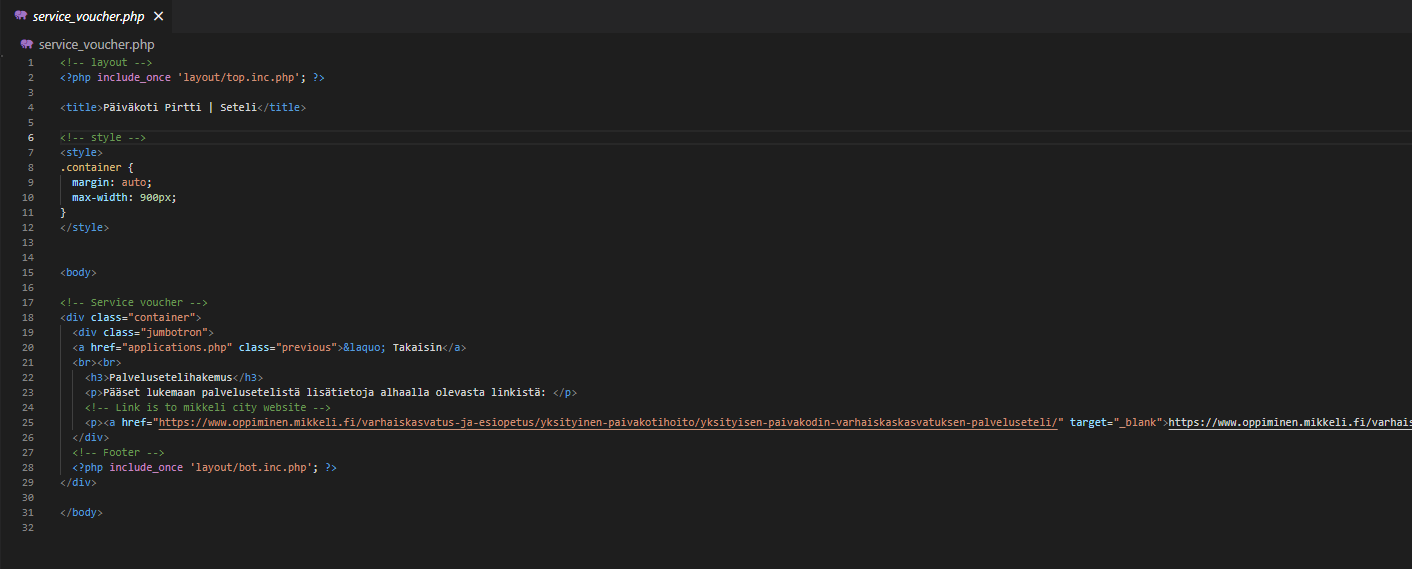
applicationReaded.php

Text

Description automatically generated

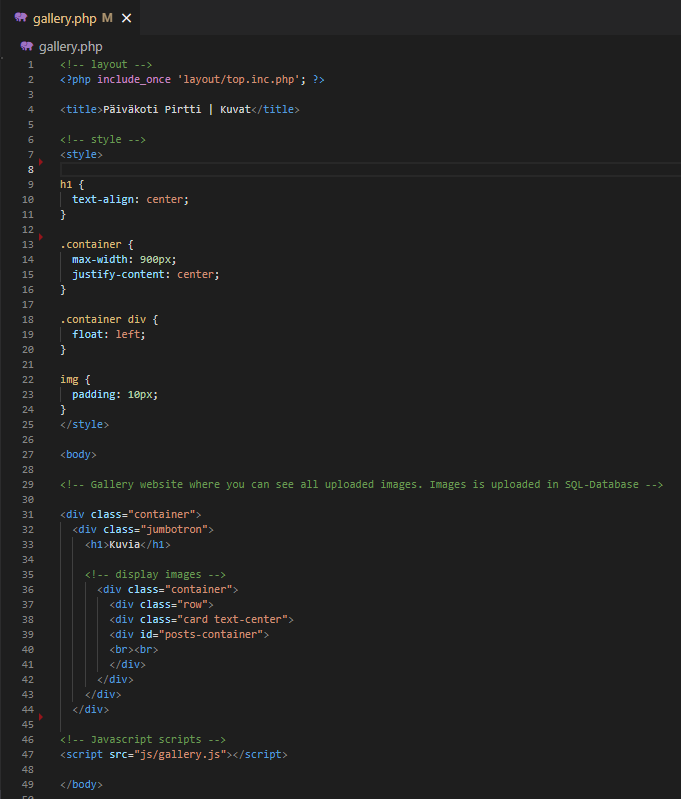
* Hakemuksen kuittaaja pääsee sähköpostin linkkiä painamalla sivulle jossa on kerrotaan, että hakemuksen tekijälle on lähetetty sähköpostia hakemuksen hyväksynnästä.

service\_voucher.php



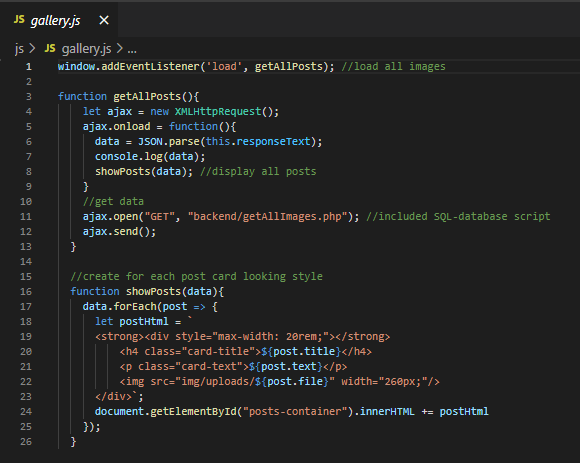
* Lisätään yläosa.
* Vaihdetaan containerin koko.
* Takasin painike jolla pääsee takaisin hakemussivustolle.
* Otsikko ja tekstiä.
* Linkki jolla pääsee mikkelin kaupungin sivustolle, jossa on palvelusetelistä tietoa. Linkki avautuu uudessa välilehdessä
* Lisätään alaosa

gallery.php



* Lisätään yläosa.
* h1 otsikko keskelle.
* containerin koko muutetaan.
* container div float left kuvat siirtyy vasemmalle.
* padding määrittää kuvien väli matka toisistaan.
* Luodaan jokaiselle kuvalle oma näkyvyys div-id = post-container avulla. Post-container löytyy JS skriptistä.

js/gallery.js



* Lisätään sivulle load tapahtuma jonka avulla ladataan sivustolle funktion avulla.
* Luodaan funktio ja tehdään ajax pyyntö.
* showPost funktio lisätään getAllPosts funktioon jotta saadaan jokaiselle postaukselle näkymä.
* post.title = näyttää otsikon postaukselle.
* post.text = näyttää tekstin postaukselle.
* post.file = näyttää kuvan postaukselle.
* ajax.open GET-parametrillä otetaan skriptin avulla tietokannasta postaukset.

backend/getAllImages.php



* Lisätään tietokanta yhteys.
* Luodaan SELECT-lauseke jolla valitaan kaikki (\*) gallery\_image taulusta.
* Suoritetaan lauseke.
* fetchAll(PDO::FETCH:ASSOC); avulla saadaan kaikki data.

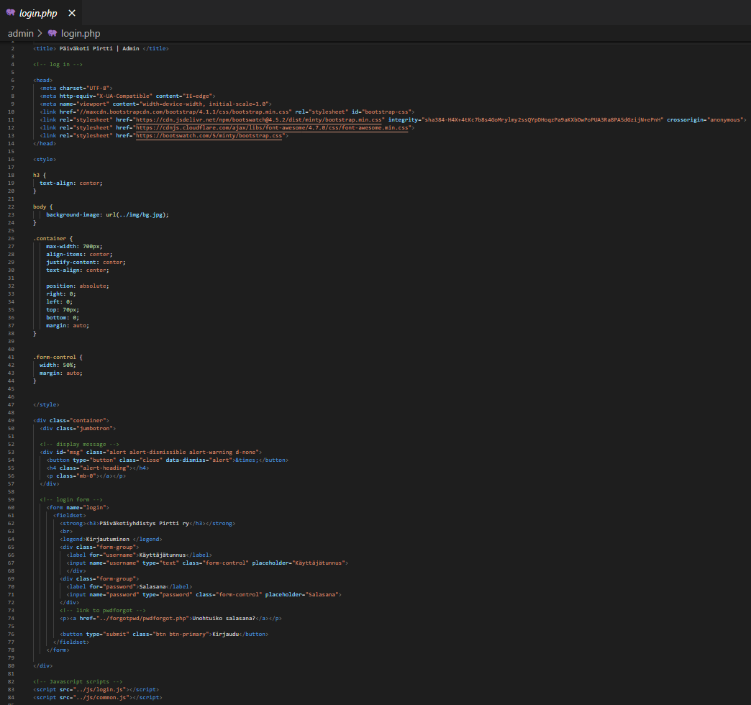
**Tiedonhallinta sivu**

js/common.js



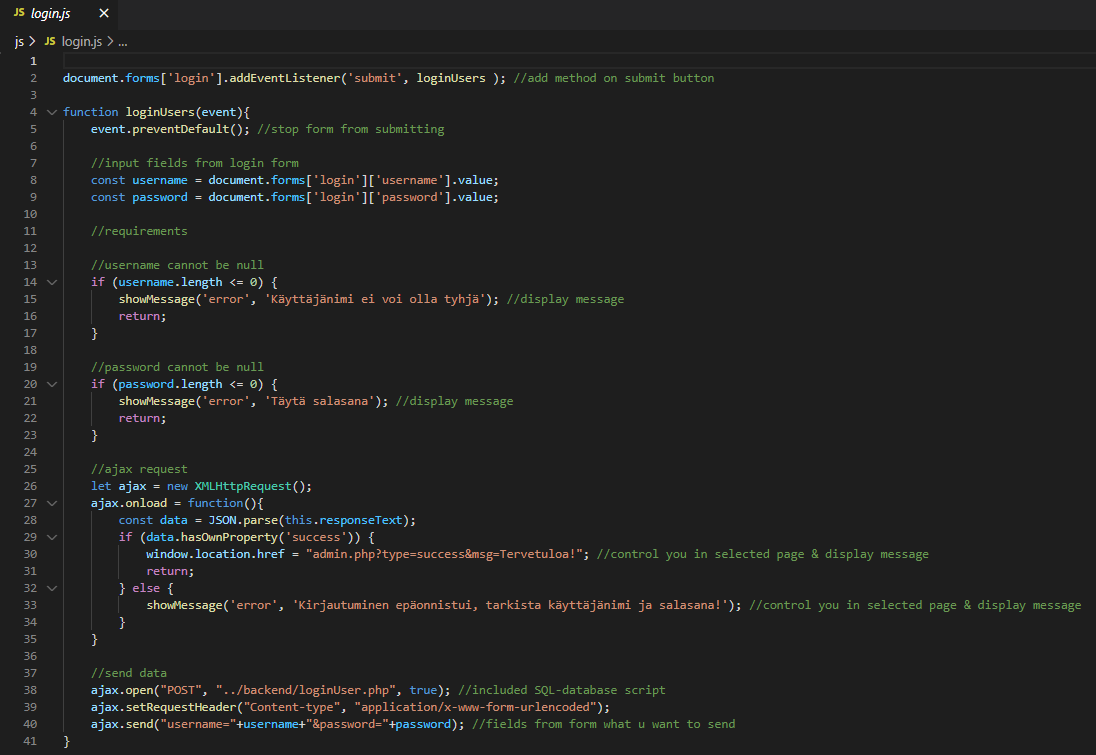
* Näyttää error & success viestit sivulla.

admin/login.php



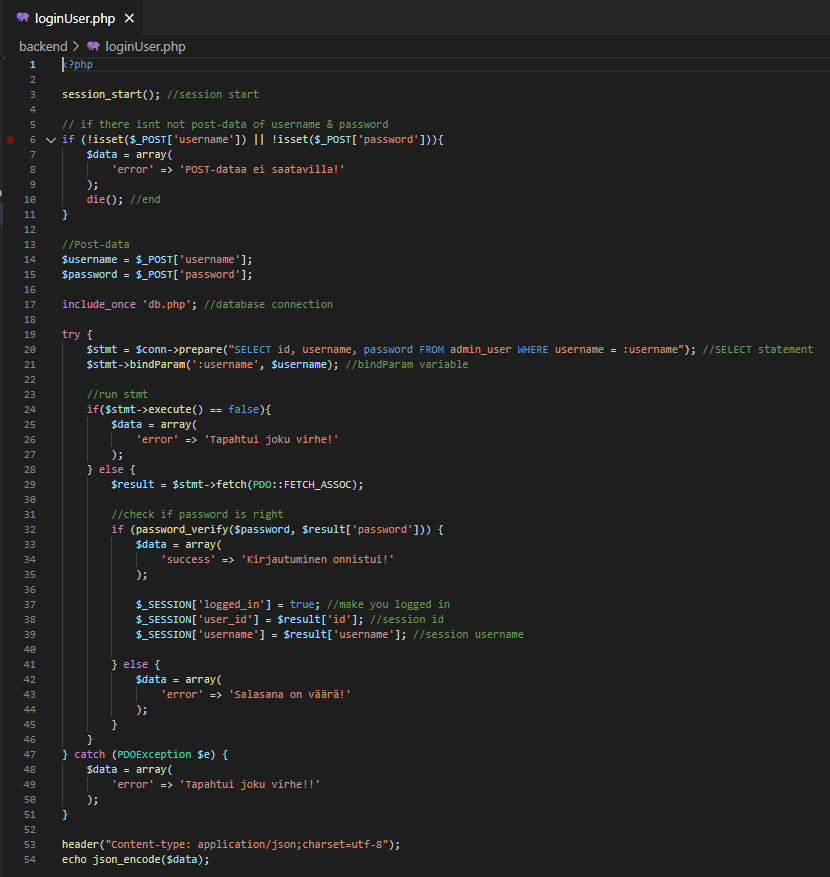
* Lisätään tarvittavat bootstrap skriptit.
* Containerin sisään on lisätty div-id=msg, joka näyttää common.js error ja success viestit.
* Sisään kirjautumis formi.
* Formin sisälle lisäsin ”unohtuiko salasana” linkin jolla voi käydä vaihtamassa salasanan sähköpostin avulla.
* Submit painikkeeseen on tehty JS:ssällä tapahtuma käsittelijä.
* Lisätään JS-skriptit (login.js, common.js) js kansiosta.

js/login.js



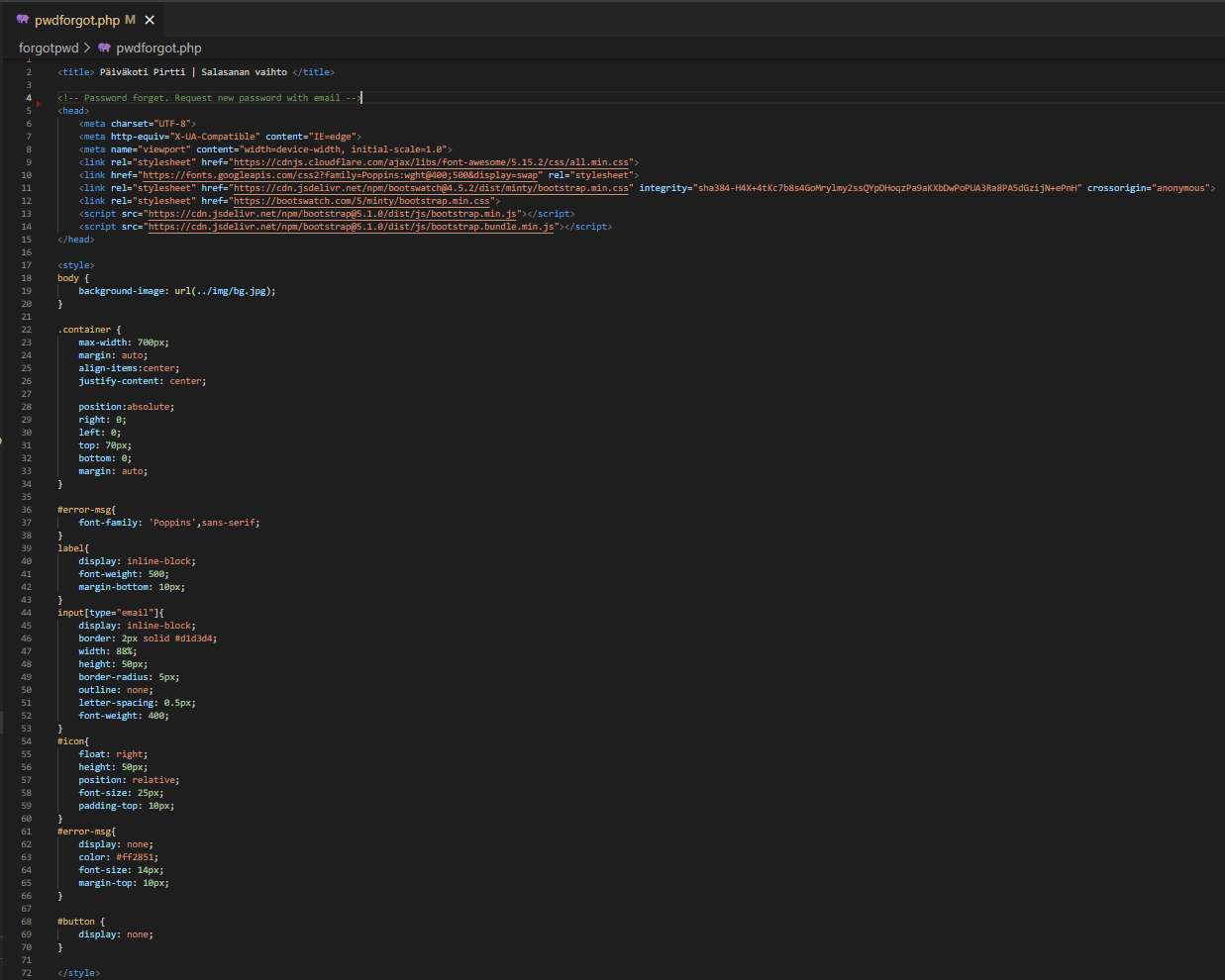
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on kaikki formin kentät ja otetaan ne talteen.¨
* Chekataan formista seuraavat asiat :
* Onko käyttäjänimi kenttä tyhjä.
* Onko salasana kenttä tyhjä.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

Backend/loginUser.php

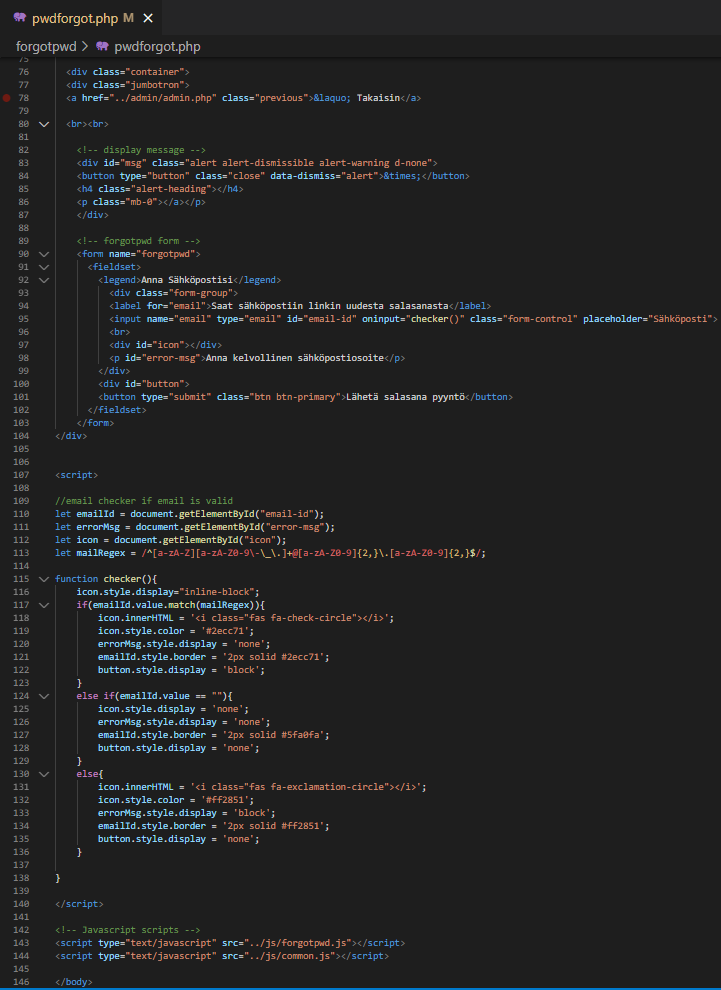


* Session alotetaan.
* Katsotaan onko username & password dataa.
* Merkataan POST-data.
* Liitetään tietokantayhteys.
* Luodaan SELECT-lauseke id, username, password jossa on username = tietokannan username.
* Suoritetaan lauseke.
* Katsotaan onko salasana sama kuin tietokannassa.
* Jos onnistuu session näyttää että olet kirjautunut.

forgotpwd/pwdforgot.php

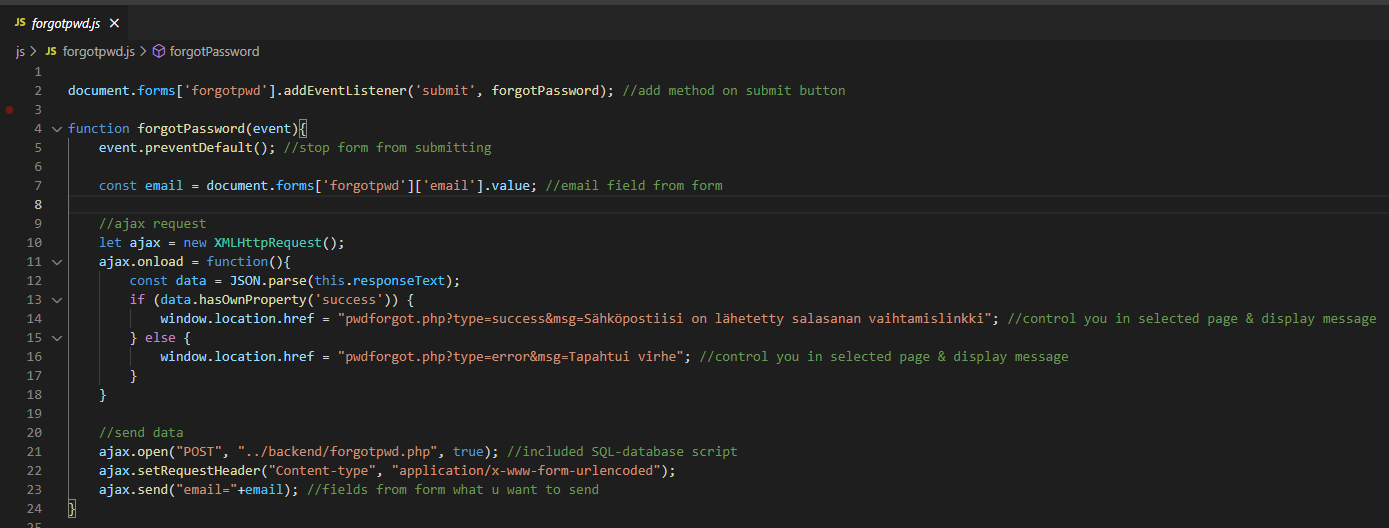


* Lisätään tarvittavat bootstrap skriptit.
* Lisätään taustakuva ja vaihdetaan containerin kokoa ja näytetään se tietyssä paikassa.
* Sähköpostin vahvistamiseen liittyviä tyylejä johon liittyy seuraavat asiat error viesti, reunat (värit), fontin koko, painikkeen piilotus.



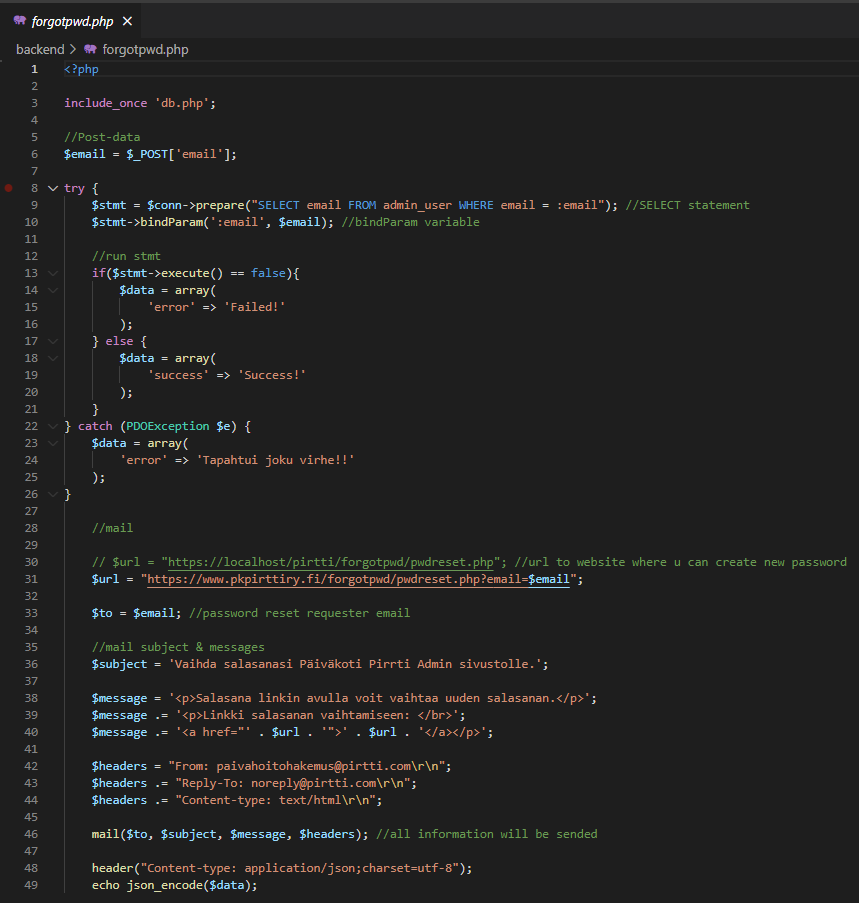
* Containerin sisään on lisätty div-id=msg, joka näyttää common.js skriptistä error ja success viestit.
* Forgotpwd formi.
* Submit painikkeeseen on tehty tapahtuman käsittelijä JS skriptillä.
* Lisäsin JS-skriptillä sivulle sähköpostin chekkaajan jos sähköposti on oikeenlainen näyttää punaisia reunoja syöttökentässä ja ristiä iconia jos sähköposti ei ole oikeenlainen ja vihreää reunoja syöttökentässä ja hyväksyntä merkintä syöttökentän alakulmassa kun sähköposti on oikeenlainen. Submit button tulee esiin vain jos sähköposti on oikeenlainen, muuten submit painiketta ei näy sivulla. Tähän liittyi tämän sivun osa CSS skriptistä.
* Liitetään pari JS skriptiä.

js/forgotpwd



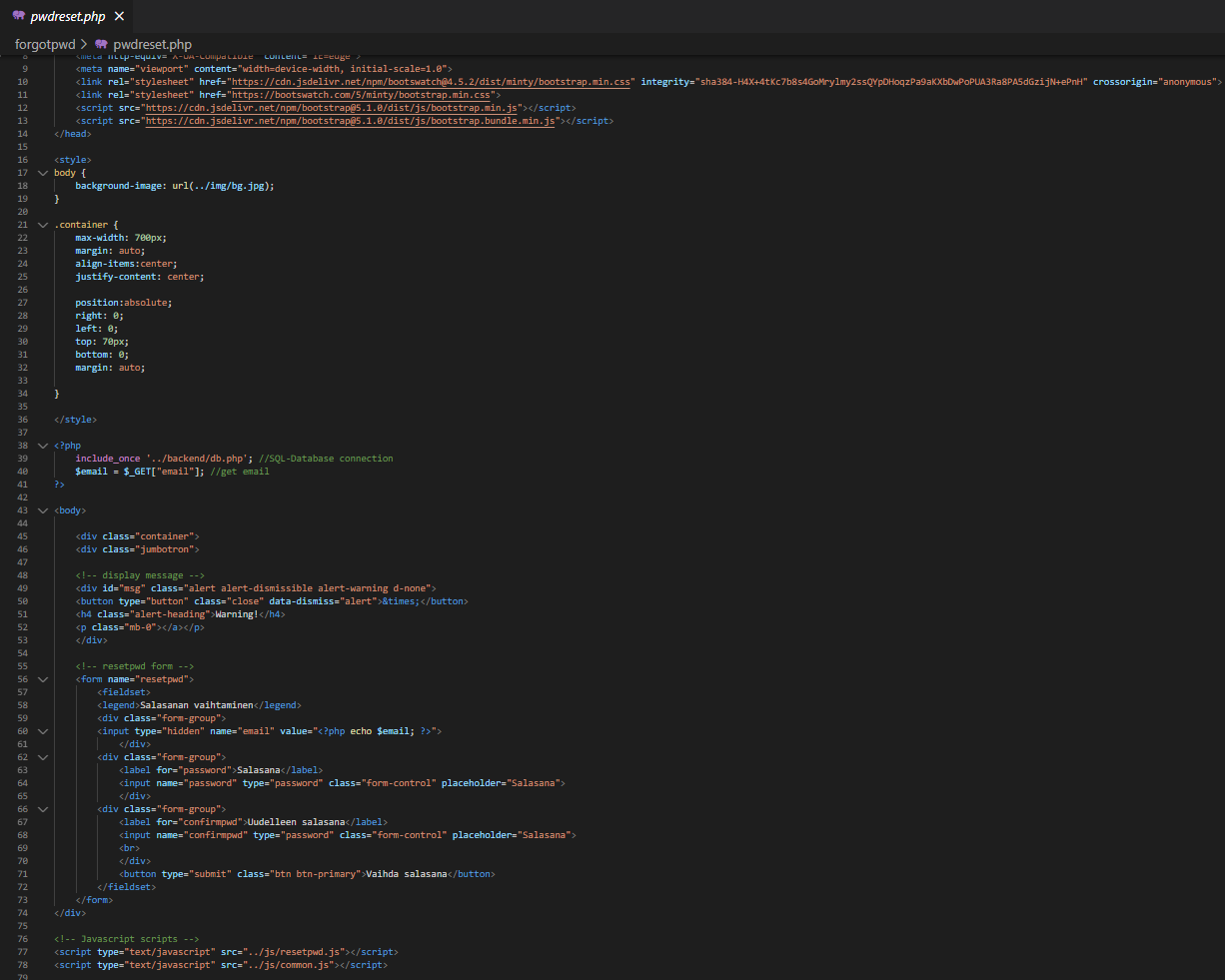
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on kaikki formin kentät ja otetaan ne talteen.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

backend/forgotpwd.php



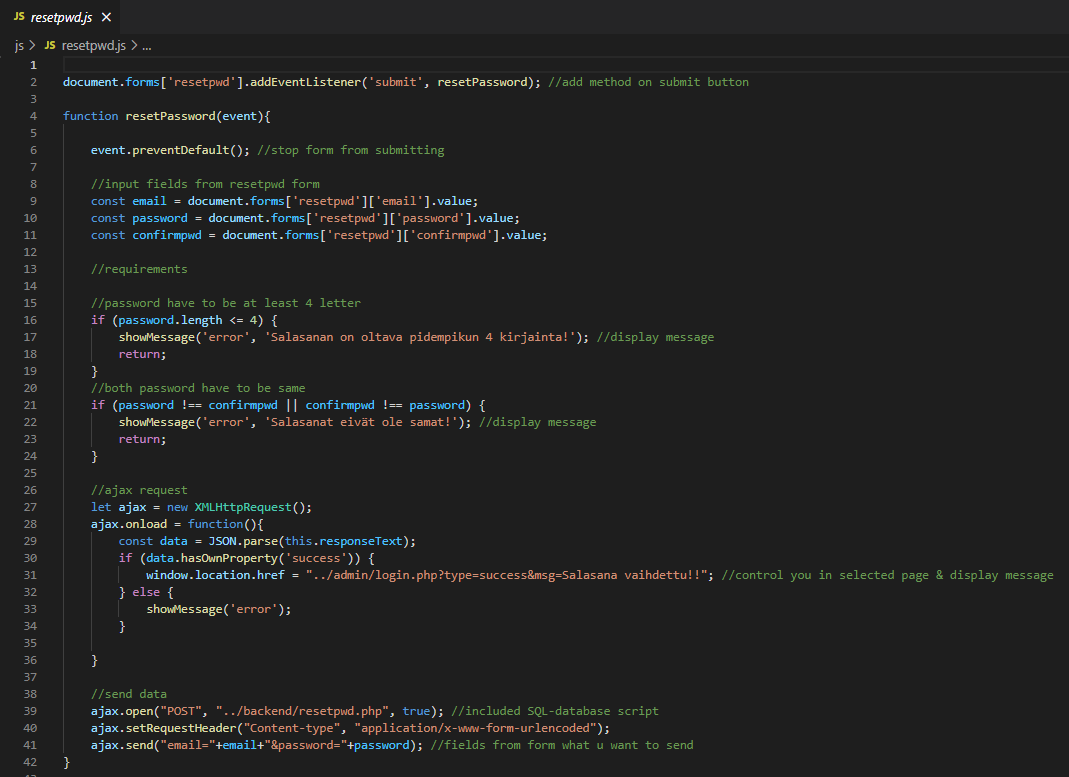
* Lisätään tietokanta yhteys.
* POST data
* Luodaan SELECT-lauseke jolla valitaan tietty sähköposti admin\_user taulusta.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.
* Salasanan vaihtamis pyyntäjälle tulee sähköpostiin sähköposti jossa on linkki salasanan vaihtamiseen.

forgotpwd/pwdreset.php



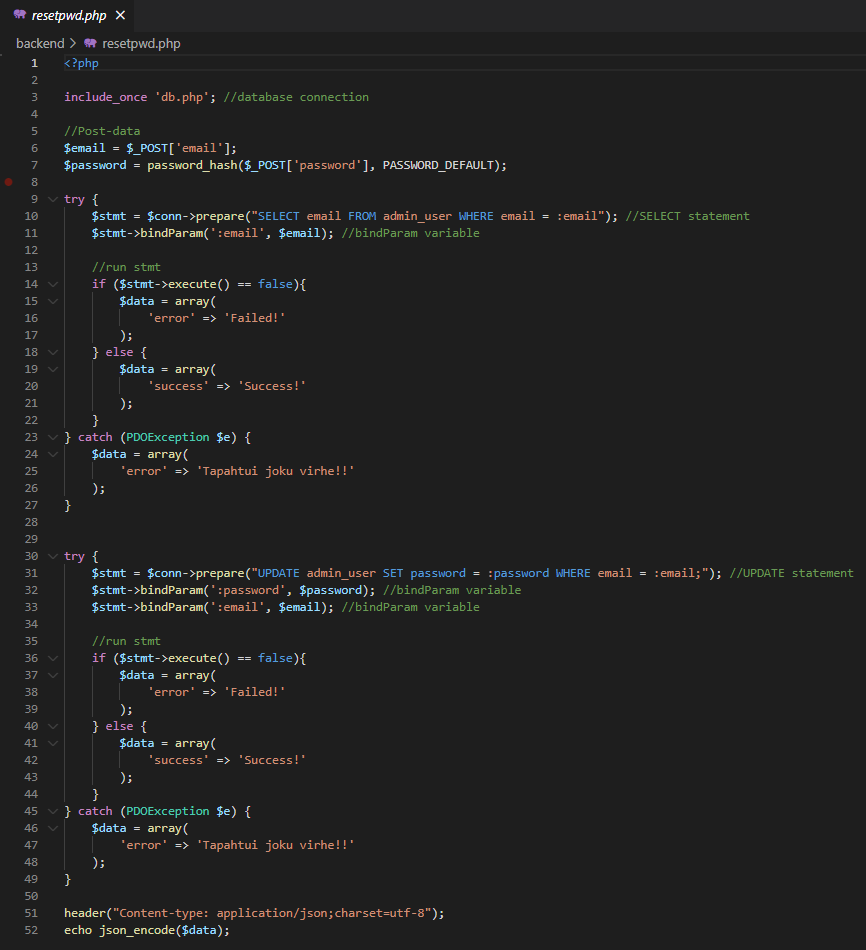
* Lisätään tietokantayhteys saadaksemme tietokannasta sähköposti.
* Containerin sisään on lisätty div-id=msg, joka näyttää common.js error ja success viestit.
* Forgotpwd formi. Formiin on lisätty hidden kenttä jossa on kyseisen käyttäjän sähköposti jotta voidaan kohdistaa sähköpostin avulla oikeaan riviin tietokannassa ja vaihtamaan salasana sähköpostin käyttäjälle.
* Submit painikkeeseen on tehty tapahtuman käsittelijä JS skriptillä.
* Liitetään pari JS skriptiä.

js/resetpwd.js



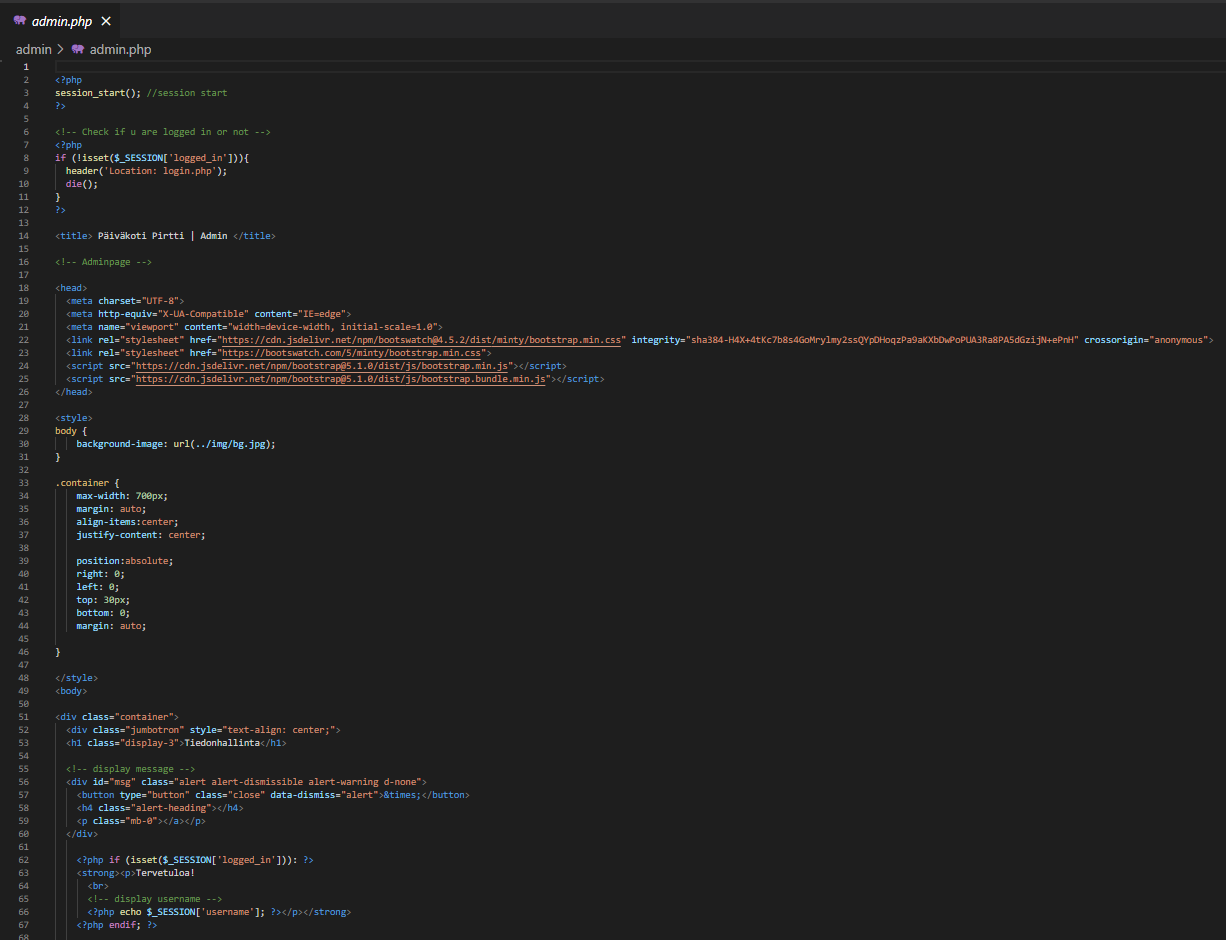
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on kaikki formin kentät ja otetaan ne talteen.
* Chekataan onko salasanassa enemmän kun 4 kirjainta ja onko salasanat samat.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

backend/resetpwd.php

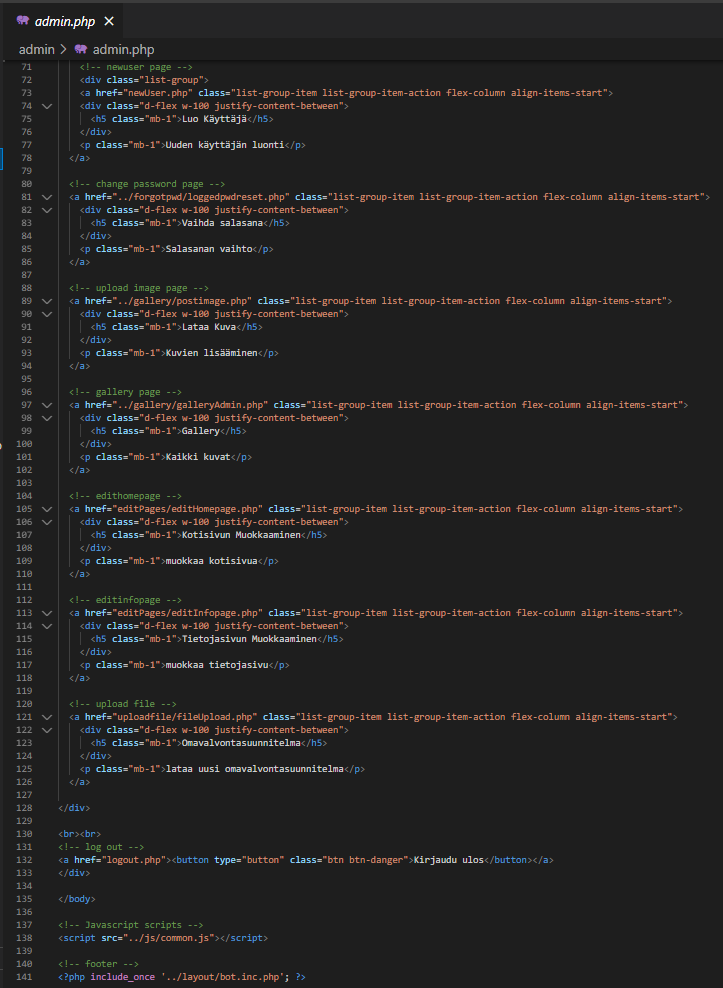


* Lisätään tietokanta yhteys.
* POST data (salasana muutetaan hash koodiksi turvallisuuden vuoksi, ettei muut tiedä salasanaa)
* Luodaan SELECT-lauseke jolla valitaan tietty sähköposti admin\_user taulusta.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.
* Luodaan UPDATE-lauseke, jolla päivitetään salasana uudeksi sähköpostin avulla.
* Suoritetaan lauseke ja katsotaan tuleeko erroreita.

Admin/admin.php

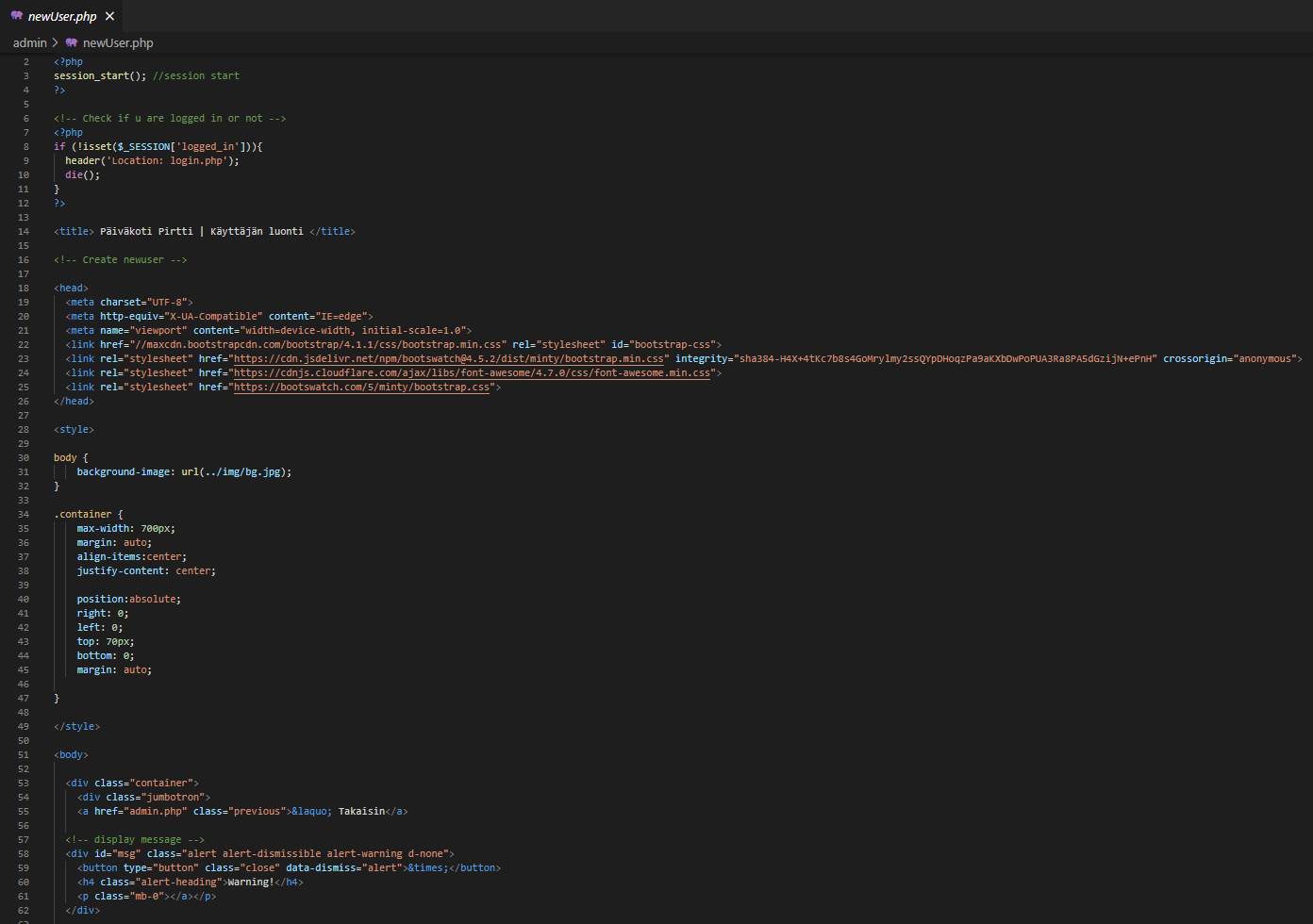


* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle vai et, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt
* Otsikko
* Error & success ilmotuksien näyttäminen
* Näytetään sivulla Tervetuloa jos olet kirjautuneena ja sinun käyttäjätunnus.

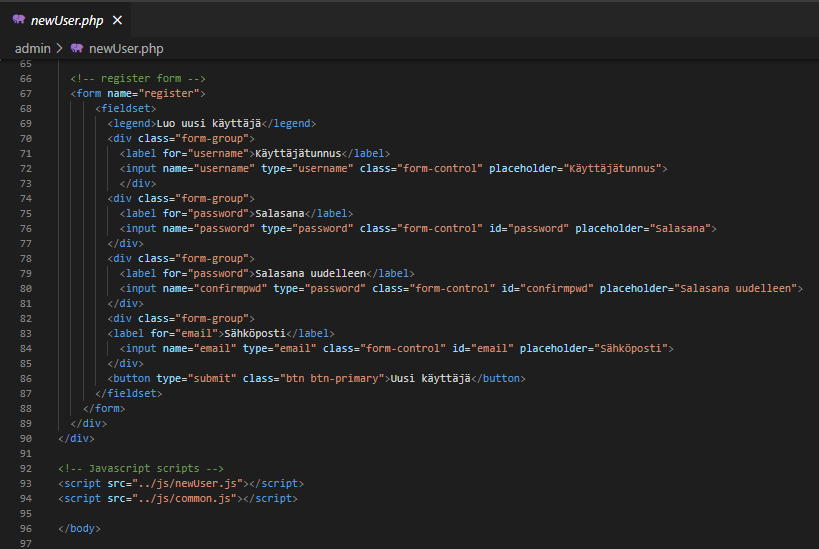


* Valikko tyyppinen listaus linkeistä josta pääsee tiettyyn linkkiin.
* Tiedonhallinta sivussa seuraavat toiminnot:
* Uuden käyttäjänluonti
* Salasanan vaihtaminen
* Kuvien lisääminen
* Galleria (kaikki lisäämät kuvat)
* Muokkaa etusivu (nettisivu)
* Muokkaa tietojasivu (nettisivu)
* Omavalvontasuunitelman lataaminen (word tiedosto)
* Kirjaudu ulos painike.
* Lisäätään JS skripti.
* Alaosa (footer).

Admin/newuser.php



* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle vai et, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt
* Takaisin painike tiedonhallinta sivulle
* Error & success ilmotuksien näyttäminen

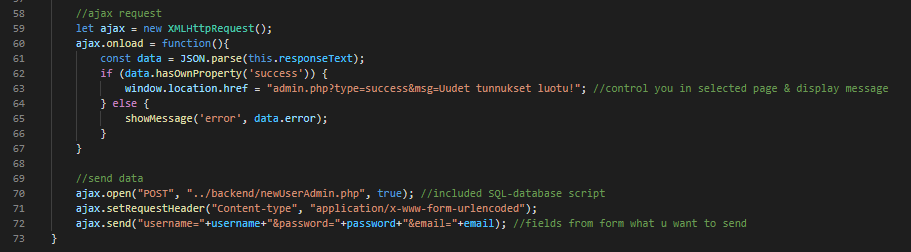


* Register formi.
* Submit nappulalle on tehty tapahtumankäsittelijä JS-skriptin avulla.
* Lisätään JS-skriptit.

js/newUser.js

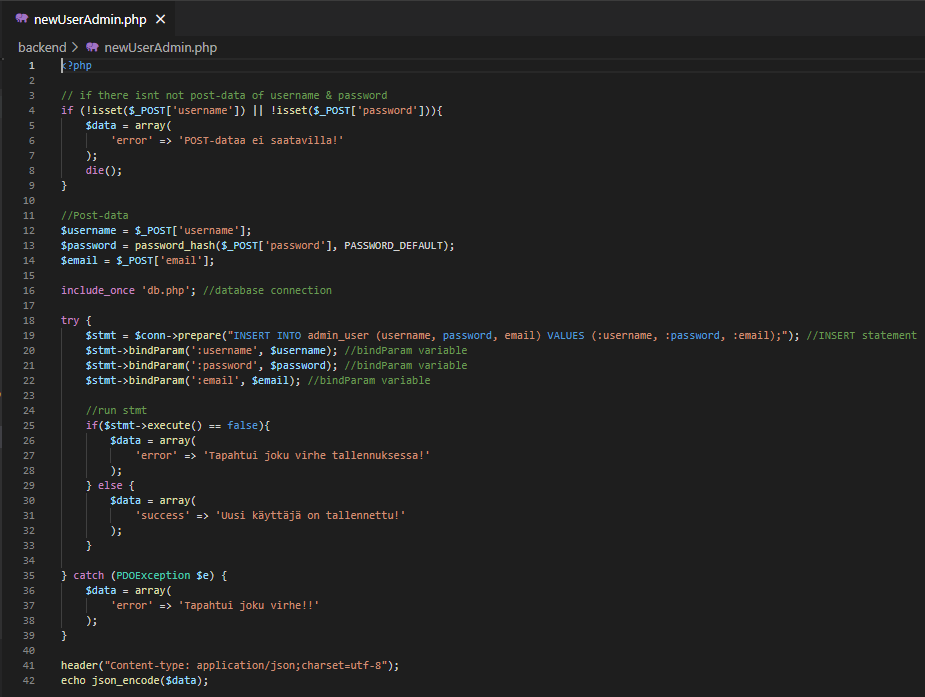


* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on kaikki formin kentät ja otetaan ne talteen.
* Chekataan seuraavat asiat :
* Onko käyttäjätunnus kenttässä mitään.
* Onko käyttäjätunnus kentässä enemmän kun 4 kirjainta.
* Onko salasana kenttässä mitään.
* Onko salasana kentässä enemmän kun 4 kirjainta.
* Onko uudelleen salasana kenttässä mitään.
* Onko uudelleen salasana kentässä enemmän kun 4 kirjainta.
* Onko salasanat samat.
* Sähköposti kenttä ei saa olla tyhjä.



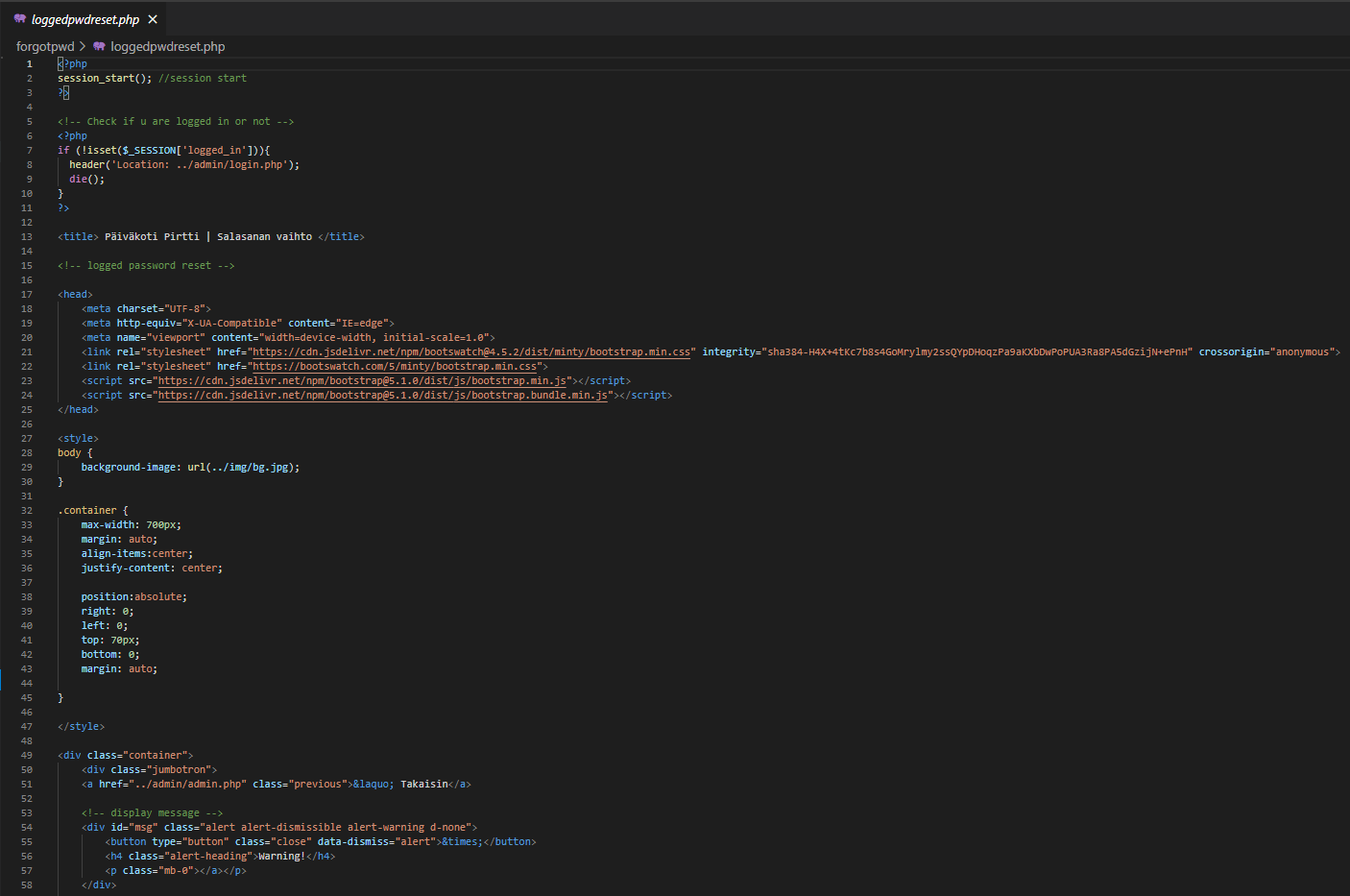
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

Backend/newUserAdmin.php



* Lisätään tietokanta yhteys.
* POST data (salasana muutetaan hash koodiksi turvallisuuden vuoksi, ettei muut tiedä salasanaa)
* Luodaan INSERT-lauseke jolla tallennetaan tiedot admin\_user tauluun.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

forgotpwd/loggedpwdreset.php

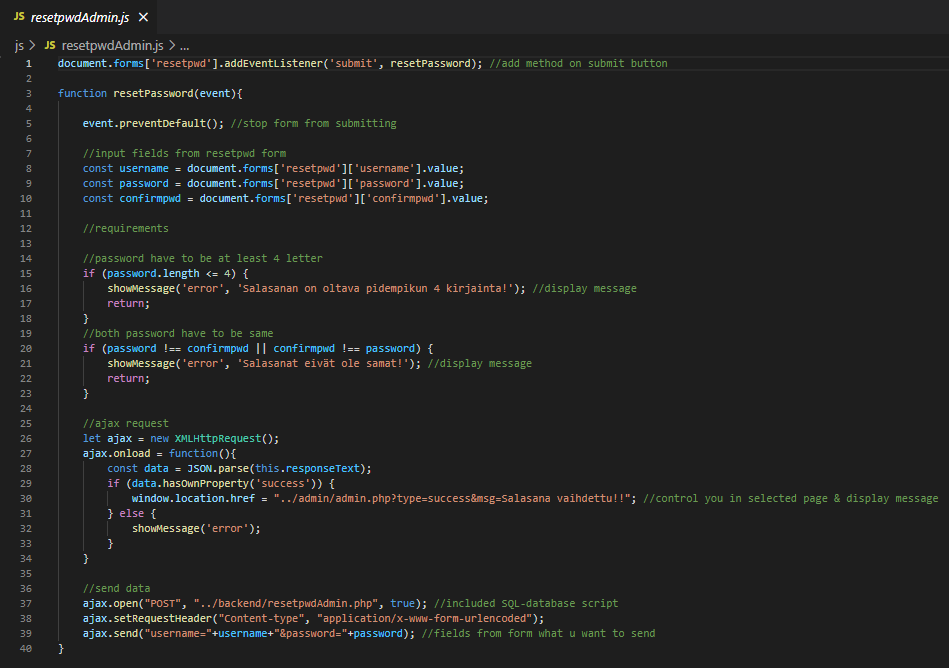


* Session alotetaan. Session avulla katsotaan oletko kirjautunut sisälle, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt
* Takaisin painike tiedonhallinta sivulle
* Error & success ilmotuksien näyttäminen



* Resetpwd formi.
* Formiin on lisätty hidden kenttä jossa on kyseisen käyttäjän käyttäjätunnus jotta voidaan kohdistaa käyttätunnukseen avulla oikeaan riviin tietokannassa ja vaihtamaan salasana.
* Lisätään JS-skriptit.

js/resetpwdAdmin



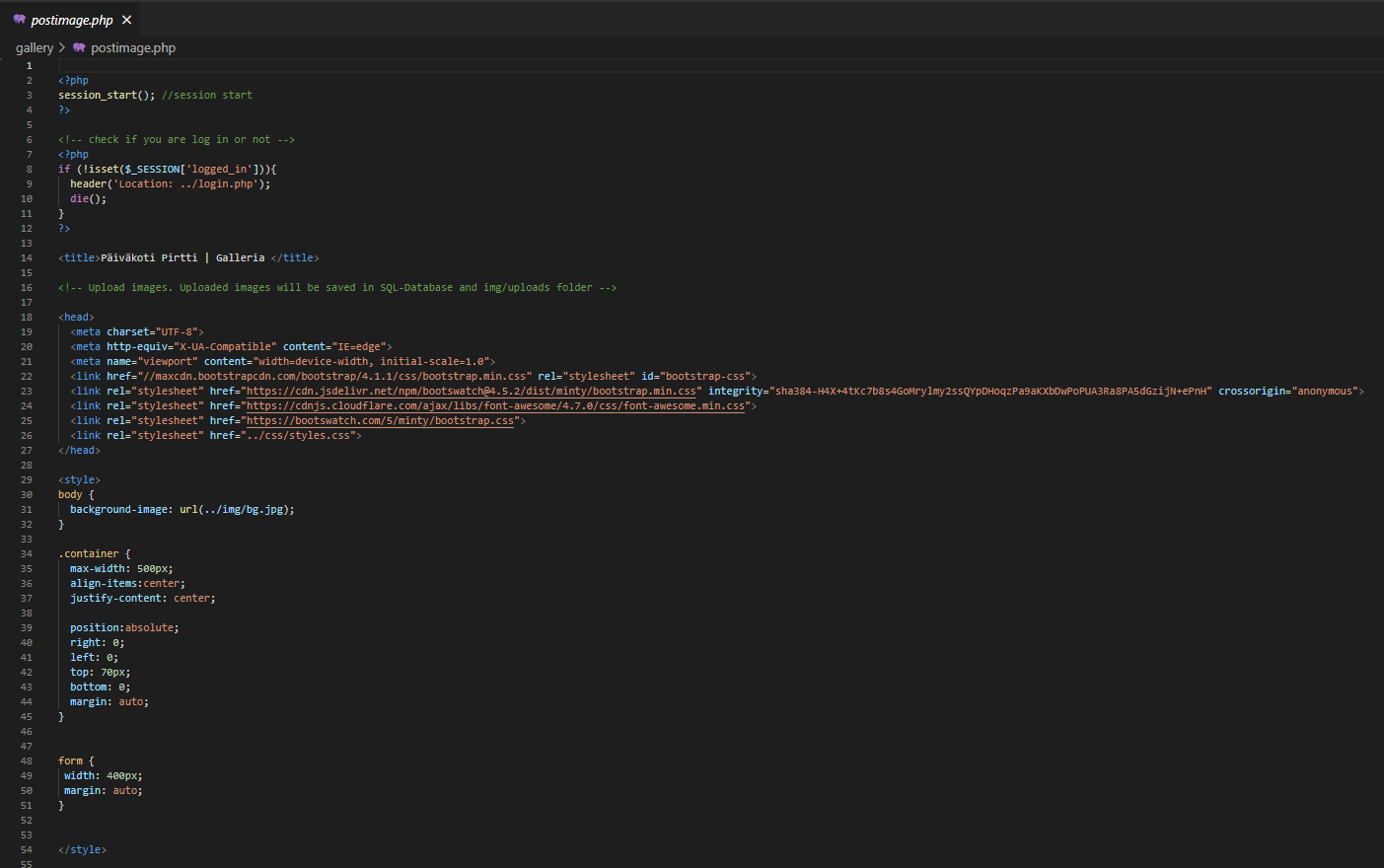
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Tehdään funktio jossa on kaikki formin kentät ja otetaan ne talteen.
* Chekataan seuraavat asiat :
* Onko salasana kentässä enemmän kun 4 kirjainta.
* Onko salasanat samat.
* Tehdään ajax kutsu jolla voidaan lähettää data eteenpäin.
* Ajax.openillä avataan skripti ja POST avulla dataa lähetetään.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

backend/resetpwdAdmin.php

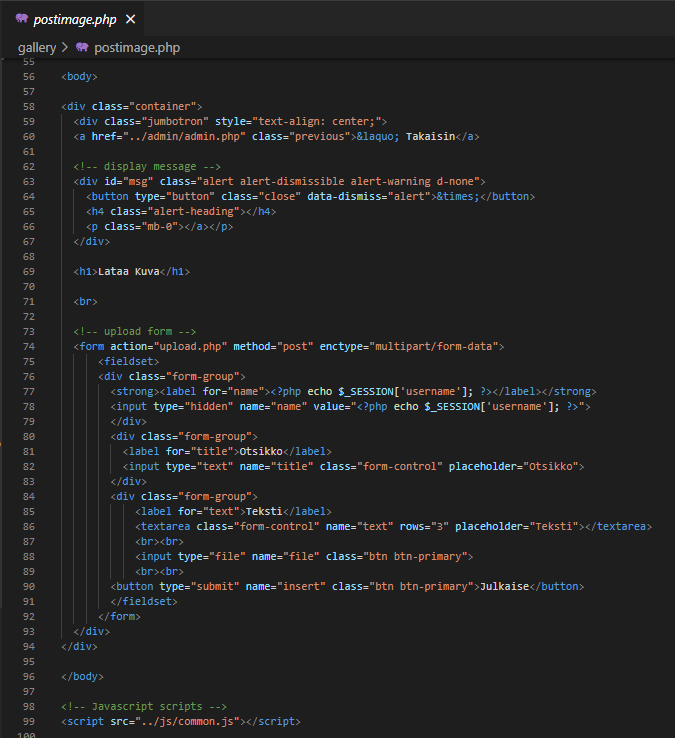


* Lisätään tietokanta yhteys.
* POST data (salasana muutetaan hash koodiksi turvallisuuden vuoksi, ettei muut tiedä salasanaa)
* Luodaan SELECT-lauseke jolla valitaan tietty sähköposti admin\_user taulusta.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.
* Luodaan UPDATE-lauseke, jolla päivitetään salasana uudeksi sähköpostin avulla.
* Suoritetaan lauseke ja katsotaan tuleeko erroreita.

gallery/postimage.php

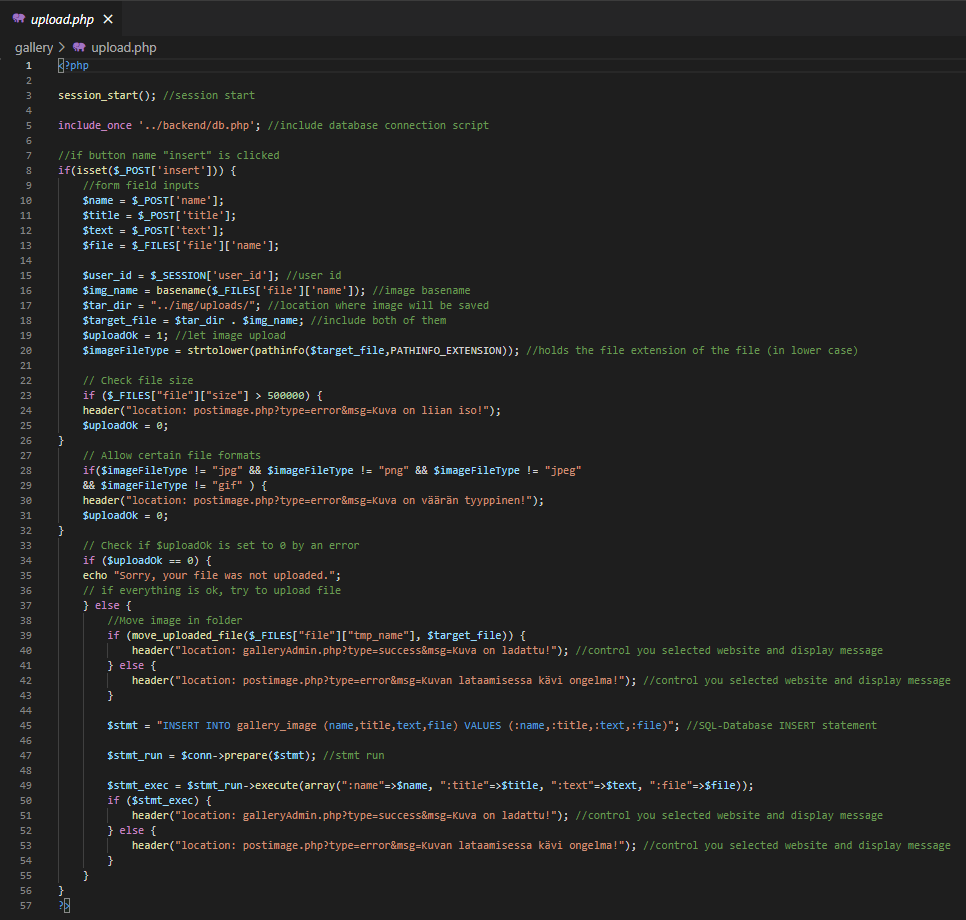


* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt.



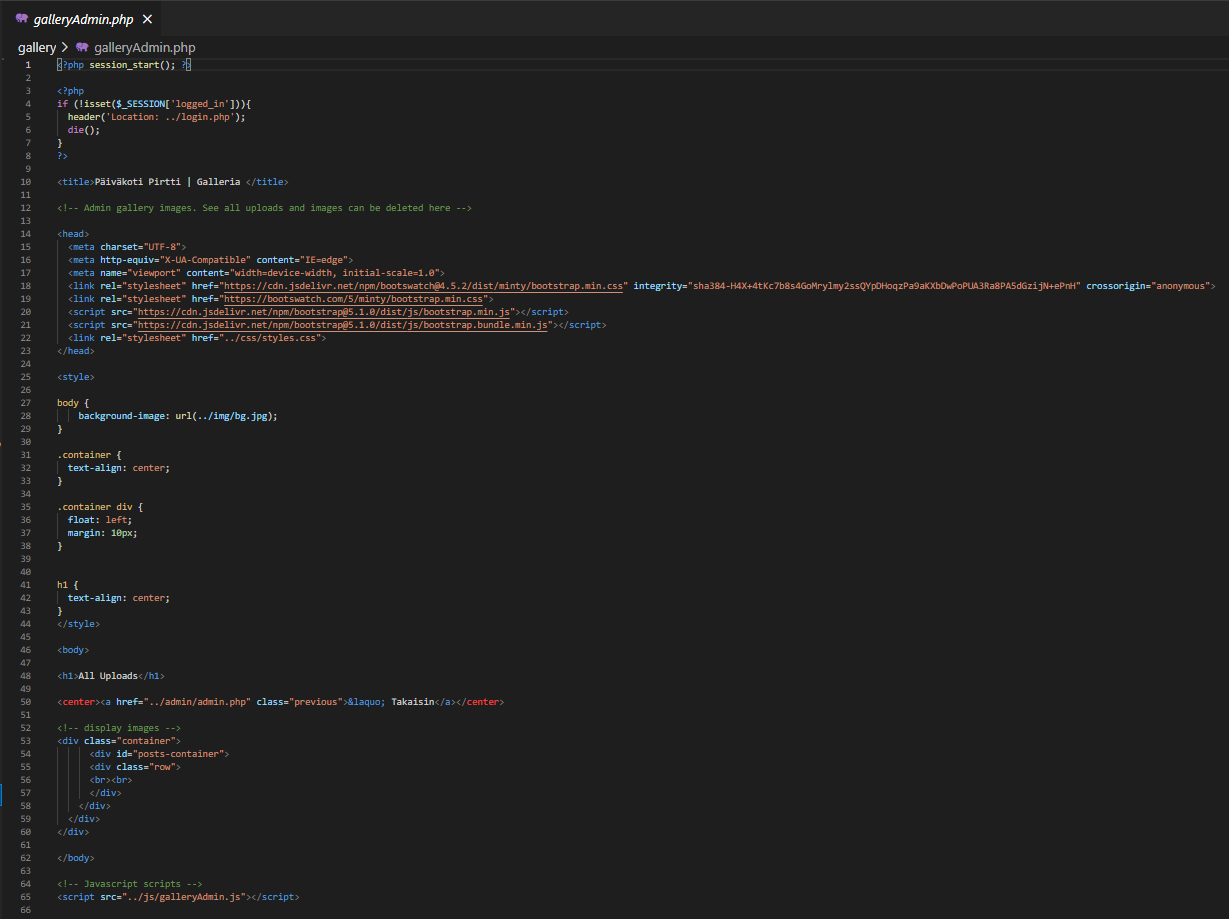
* Takaisin painike tiedonhallinta sivulle.
* Error & success ilmotuksien näyttäminen.
* Formiin on lisätty action, jotenka tämä formi toimii upload.php skriptin avulla.
* Upload formiin on lisätty tiedostotyypinen mekanismi enctype=”multipart/form-data” luo formiin painikkeen jolla voit avata oman tietokoneesi tiedostot.
* Käyttäjätunnus otetaan sessionin avulla. Tunnus tallentuu tällä tavoin samalla.
* Lisätään JS-skripti.

gallery/upload.php



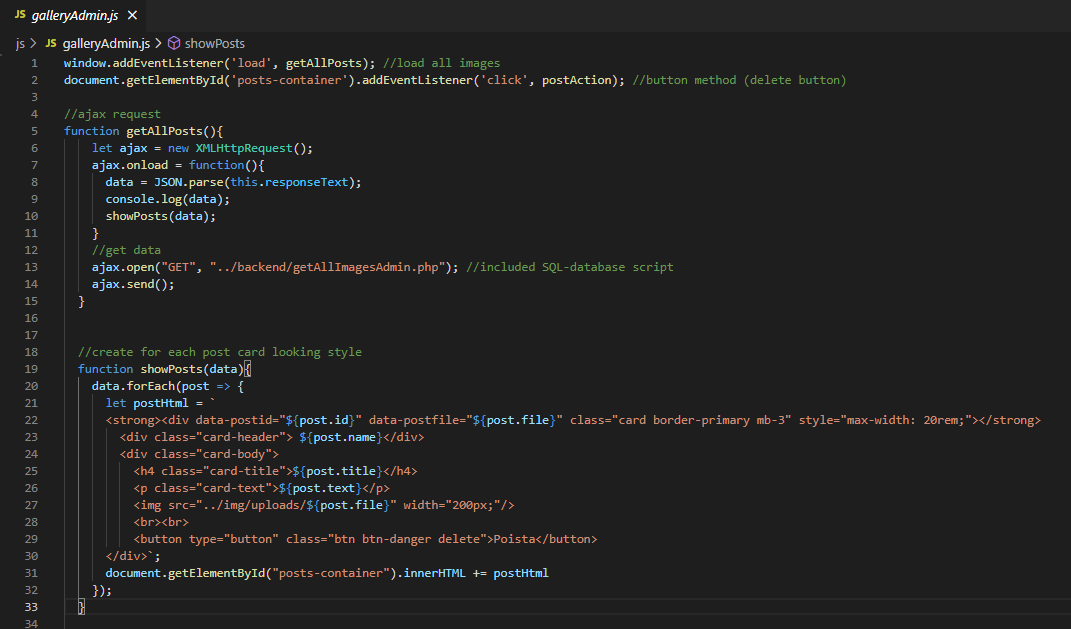
* Session alotus.
* Lisätään tietokantayhteys.
* Katsotaan jos insert painiketta on painettu.
* Otetaan formin kentät.
* $img\_name = otetaan kuvan oikeanimi.
* $tar\_dir = kuvan tallennus reitti.
* $target\_file = kuvan oikeanimi + kuvan tallennus reitti.
* $uploadOk = määrittää suoritetaanko kuvan uploadaaminen tai ei.
* Chekataan seuraavat asiat :
* Onko tiedosto yli 500000 bittiä (476MB).
* Tiedoston tyyppi (jpg, png, jpeg, gif).
* Jos uploadOk on 0 niin tulee error muuten päästään siirtämään tiedostoa.
* move\_uploaded\_file avulla siirretään kuva.
* $target\_file avulla tallennetaan kuva oikeella nimellä ja reitti johon kuva halutaan tallentaa.
* Katsotaan tuleeko erroreja.
* Muuten tallennetaan INSERT-lausekkeella gallery\_image tauluun tietokantaan tiedot.

gallery/galleryAdmin.php

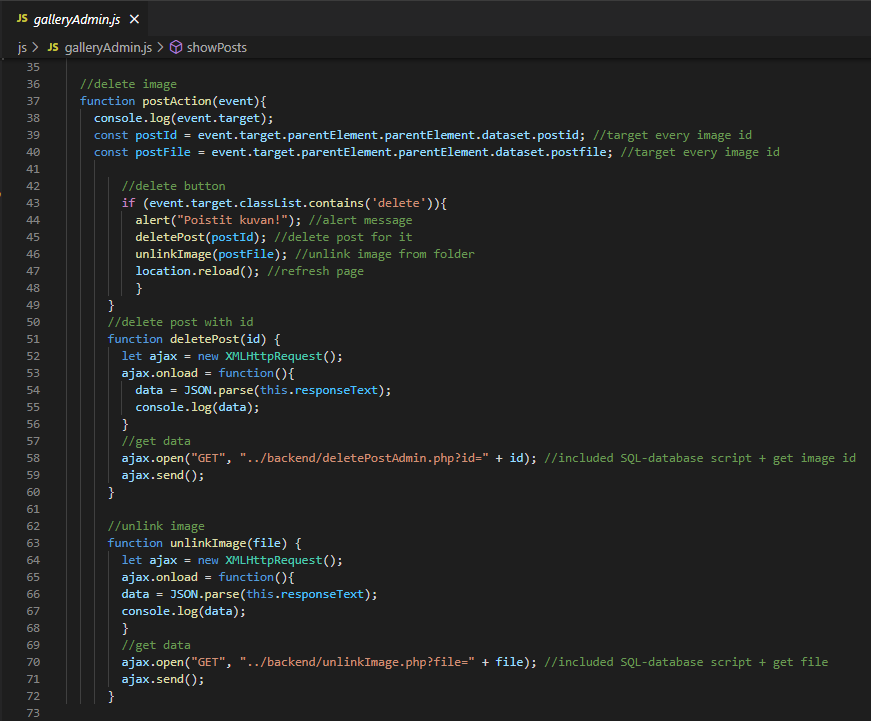


* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt.
* Takaisin painike tiedonhallinta sivustolle.
* Jokaiselle postaukselle omanäkymä div-id = post-container avulla.
* Lisätään JS-skripti.

js/galleryAdmin.js

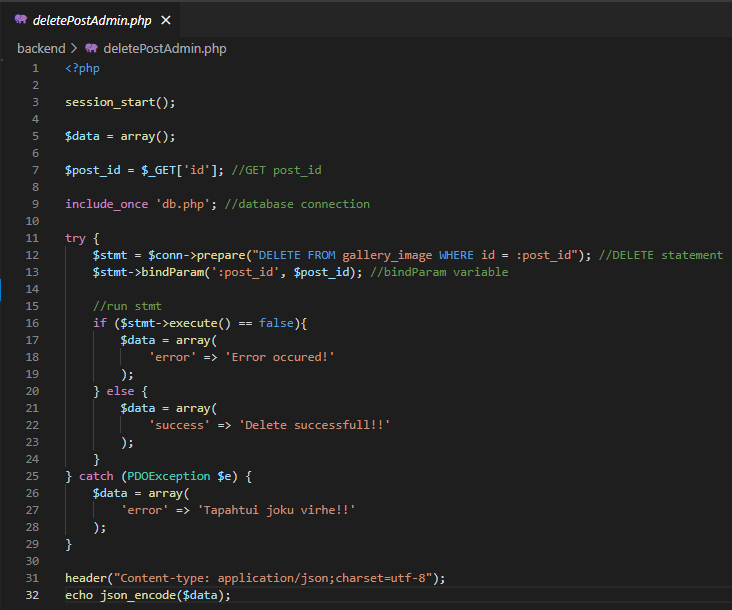


* Lisätään sivulle load tapahtuma jonka avulla ladataan sivustolle funktion avulla.
* Lisätään sivulle submit tapahtuma jonka avulla ladataan sivustolle funktion avulla.
* Luodaan funktio ja tehdään ajax pyyntö kuville.
* showPosts funktio lisätään getAllPosts funktioon jotta saadaan jokaiselle postaukselle näkymä.
* post.id = postauksen id.
* post.file = kuvan nimi.
* post.title = näytetään otsikko postaukselle.
* post.text = näytetään teksti postaukselle.
* post.file = näytetään kuva postaukselle(kuvan nimi).
* Jokaiselle postaukselle tulee ”Poista” näppäin. Poista näppäimelle tehdään oma toiminto.



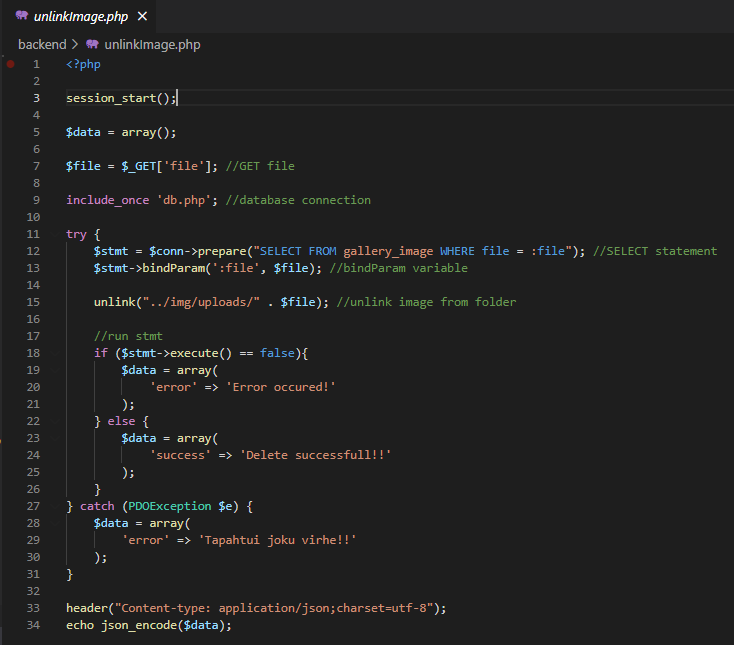
* Luodaan funktio.
* postId saadaan postauksesta (post.id). Siihen kohdistetaan event.target.parentElement.parentElement.dataset.postid avulla.
* postFile saadaan postauksesta (post.file) . Siihen kohdistetaan event.target.parentElement.parentElement.dataset.postfile avulla.
* Katsotaan onko painikkeen classissä ”delete”.
* Alertti tulee aina kun poistat kuvan.
* Luodaan kaksi funcktioo.
* deletePost(postId) jolla poistetaan kuva ID:n avulla näkyvistä sivulla ja tietokannasta.
* unlinkImage(postFile) jolla poistetaan kuva nimen avulla kansiosta.
* Tehdään molemmista ajax kutsu GET parametrillä.
* deletePost otetaan id mukaan ajax.open skriptiin ja unlinkImage otetaan file (kuvan nimi) mukaan skriptiin.

Backend/deletePostAdmin.php



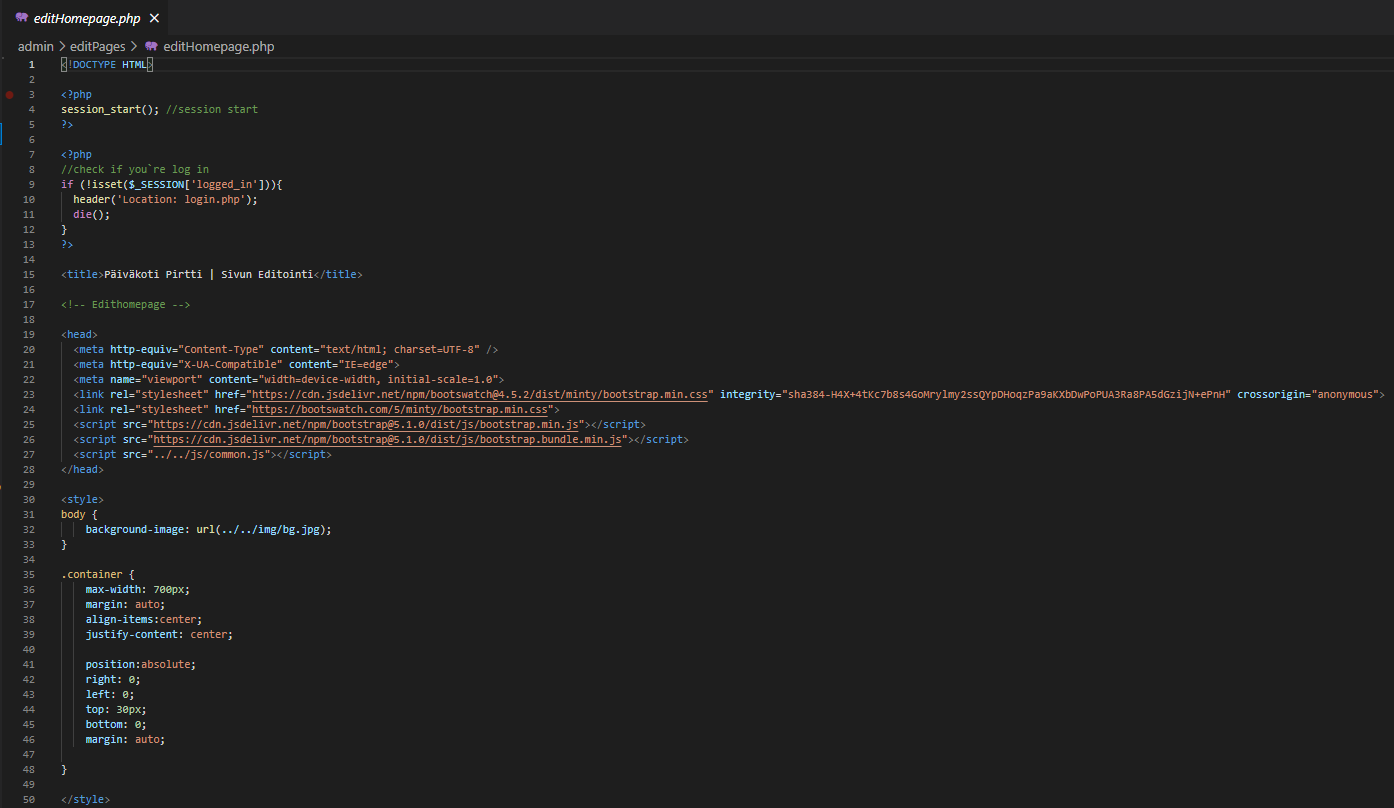
* Session alotetaan.
* Otetaan id.
* Liitetään tietokanta yhteys.
* Tehdään DELETE-lauseke jolla poistetaan gallery\_image taulusta kyseisen kuvan ID:n avulla kuva.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

Backend/unlinkImage.php

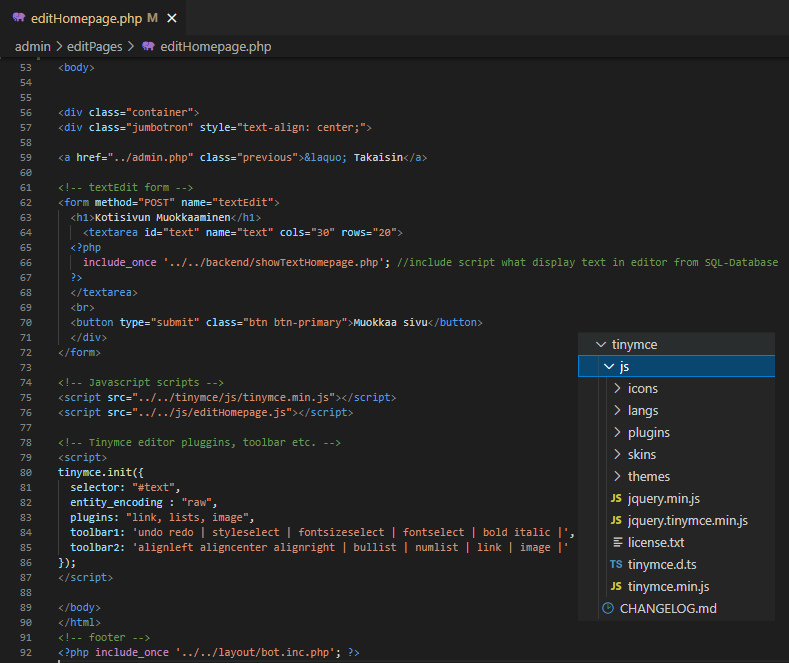


* Session alotetaan.
* Otetaan file.
* Liitetään tietokanta yhteys.
* Tehdään SELECT-lauseke jolla kohdistetaan gallery\_image taulusta kyseisen kuvan nimen avulla oikea kuva.
* unlink poistaa kuvan kyseisestä polusta.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

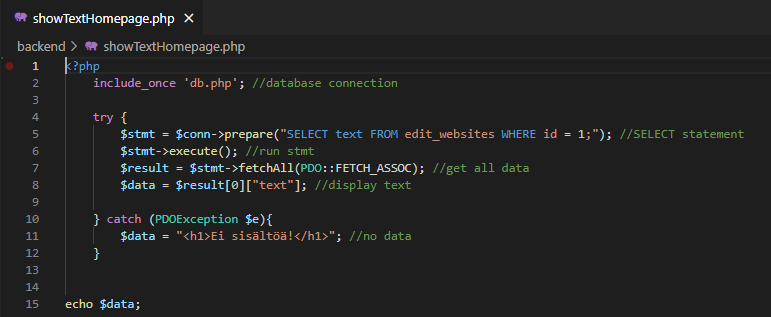
Admin/editPages/editHomepage.php



* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt.

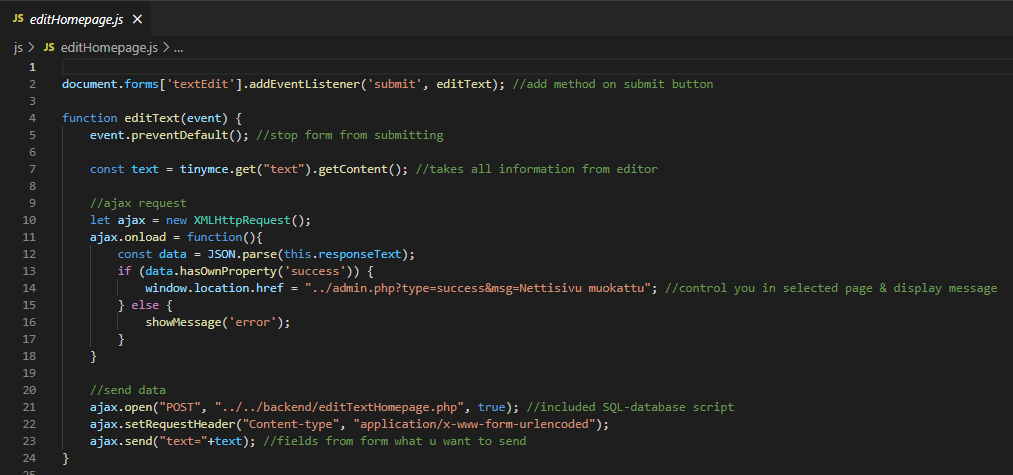


* Takaisin painike tiedonhallinta sivustolle.
* Luodaan textEdit formi. Luodaan formiin tekstieditori tinymce.
* Otetaan tämän hetkinen teksti tietokannasta.
* Submit painikkeelle on tehty tapahtumankäsittelijä JS-skriptin avulla.
* Lisätään JS-skriptit.
* Tinymce.init avulla lisätään tinymce editoriin toimintoja.



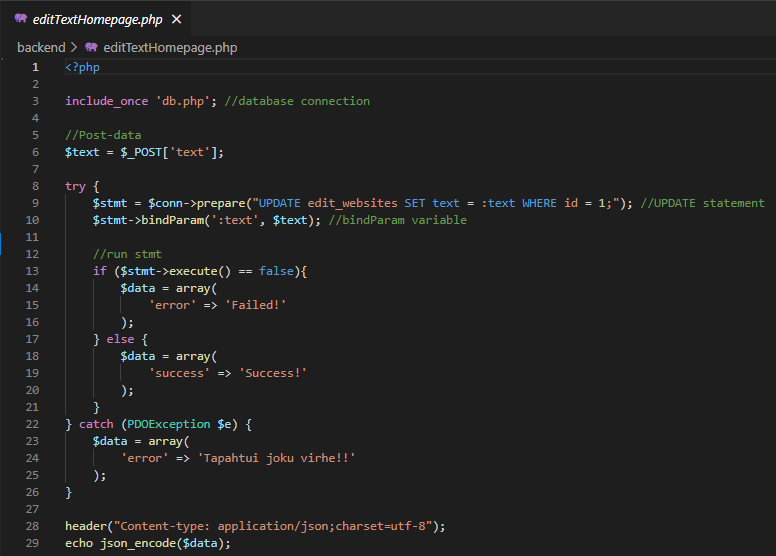
* Lisätään tietokanta yhteys.
* Luodaan SELECT-lauseke jolla kohdistetaan edit\_website taulun text riviin jonka id on 1.
* Suoritetaan lauseke.
* Otetaan kaikki data.
* Näytetään teksti osio.

js/editHomepage.js



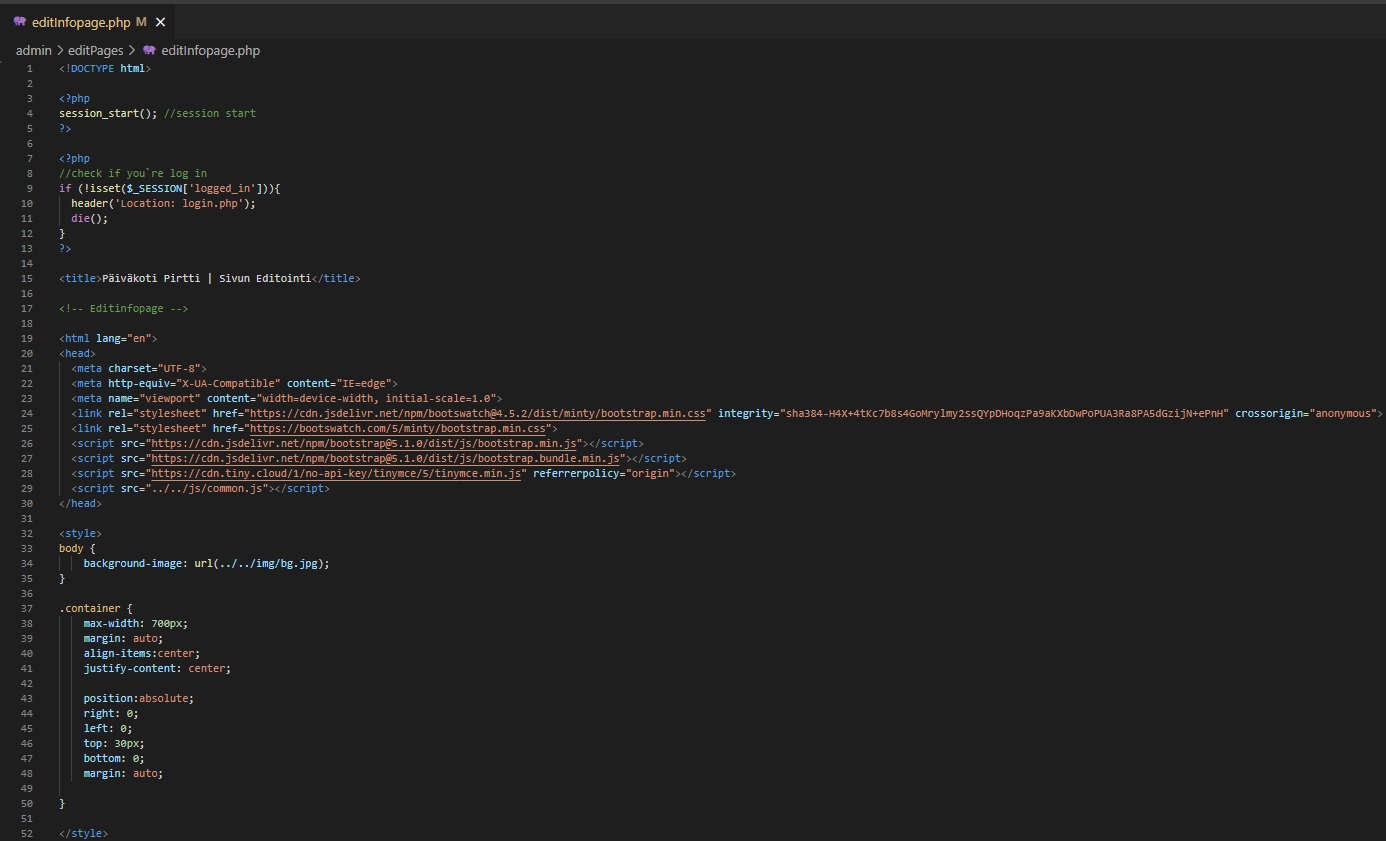
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Luodaan funktio.
* Otetaan teskti editorista.
* Tehdään ajax pyyntö.
* Ajax.openillä avataan skripti ja POST parametrillä.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

backend/editTextHomepage.php

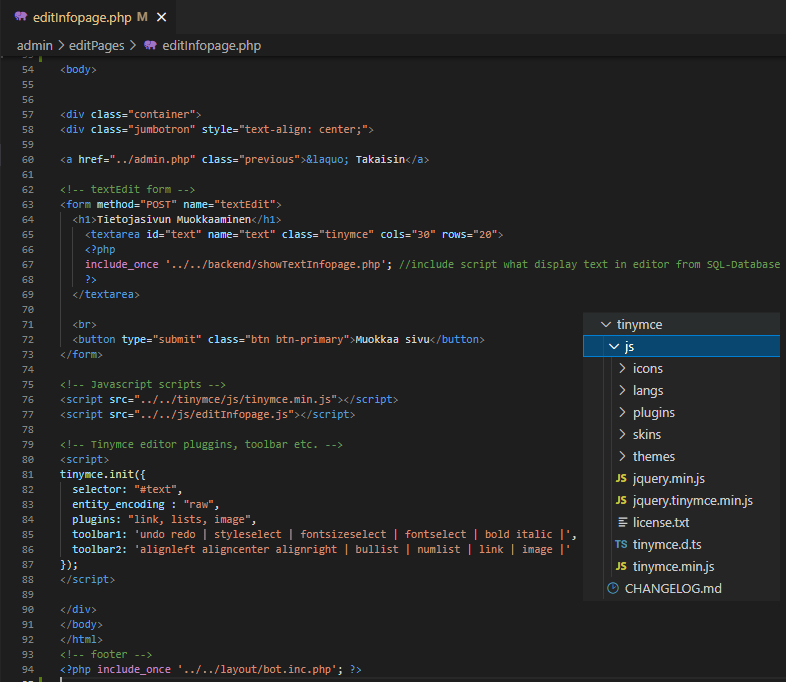


* Lisätään tietokanta yhteys.
* POST-data
* Luodaan UPDATE-lauseke jolla kohdistetaan edit\_website taulun text riviin jonka id on 1.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

admin/editPage/editInfoPage.php

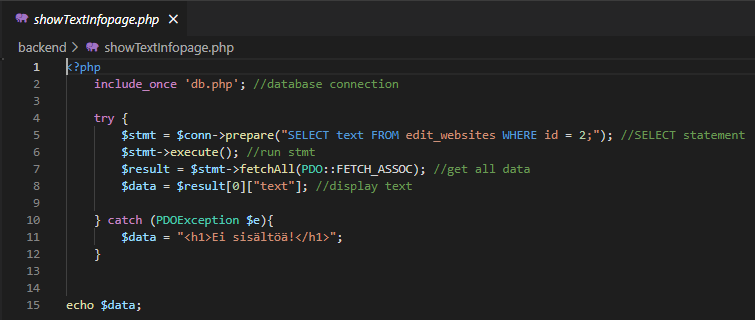


* Session alotetaan.
* Session avulla katsotaan oletko kirjautunut sisälle, muuten sinut siirretään login.php skriptiin.
* Bootstrap skriptit.
* Tyylittelyt.



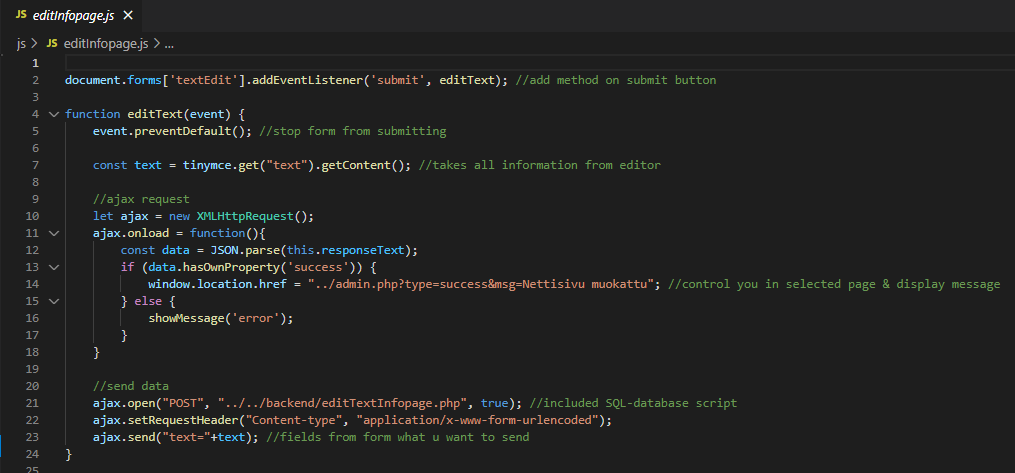
* Takaisin painike tiedonhallinta sivustolle.
* Luodaan textEdit formi. Luodaan formiin tekstieditori tinymce.
* Otetaan tämän hetkinen teksti tietokannasta.
* Submit nappulalle on tehty tapahtumankäsittelijä JS-skriptin avulla.
* Lisätään JS-skriptit.
* Tinymce.init avulla lisätään tinymce editoriin toimintoja.

backend/showTextInfopage.php



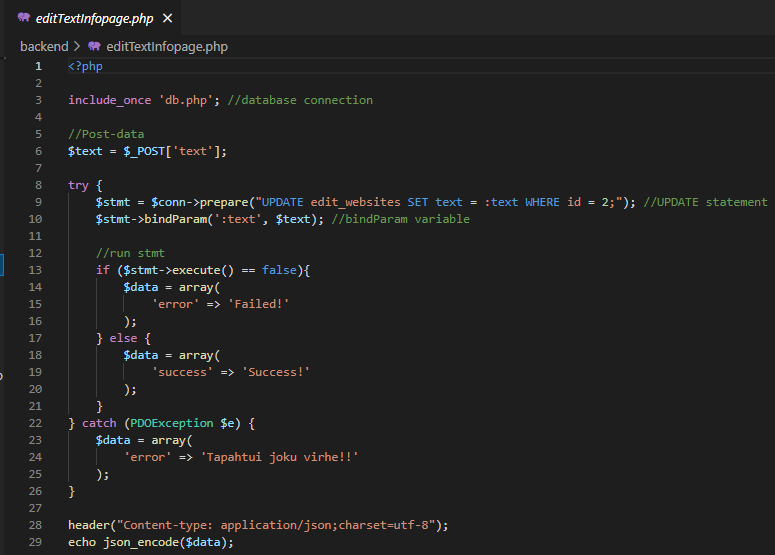
* Lisätään tietokanta yhteys.
* Luodaan SELECT-lauseke jolla kohdistetaan edit\_website taulun text riviin jonka id on 2.
* Suoritetaan lauseke.
* Otetaan kaikki data.
* Näytetään teksti osio.

js/editInfopage.js



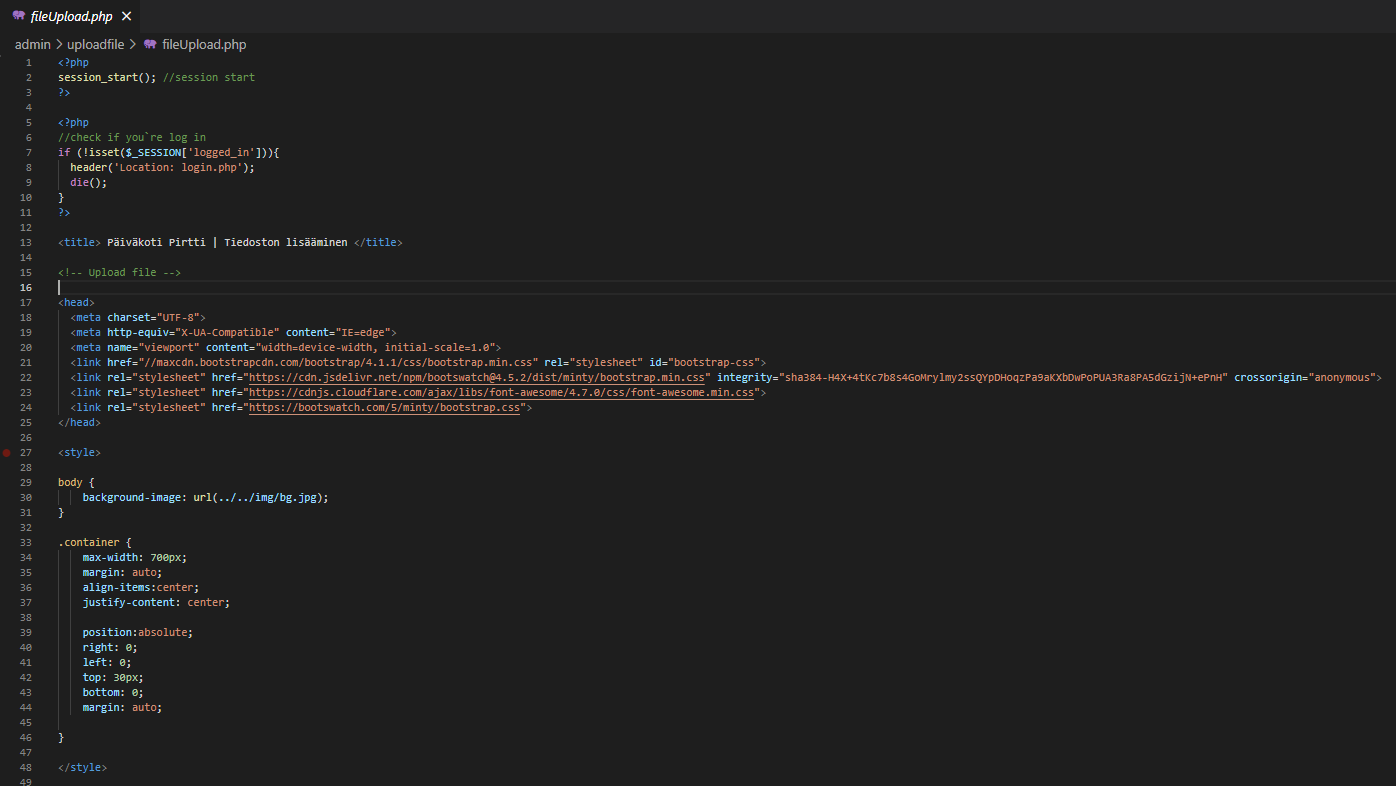
* Lisätään submit painikkeeseen tapahtumakäsittelijä, joka käsittelee formissa olevia kenttiä.
* Luodaan funktio.
* Otetaan teksti editorista.
* Tehdään ajax pyyntö.
* Ajax.openillä avataan skripti ja POST parametrillä.
* Ajax.sendillä lähetetään data ajax.open skriptiin.

backend/editTextInfopage.php

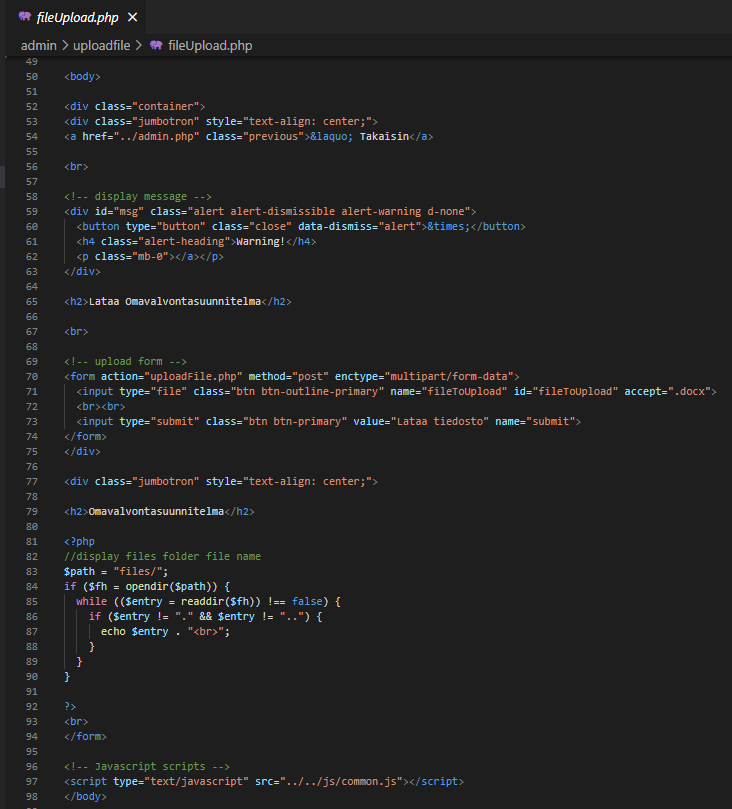


* Lisätään tietokanta yhteys.
* POST-data.
* Luodaan UPDATE-lauseke jolla kohdistetaan edit\_website taulun text riviin jonka id on 2.
* Suoritetaan lauseke ja katsotaan tuleeko erroreja.

admin/uploadfile/fileUpload.php

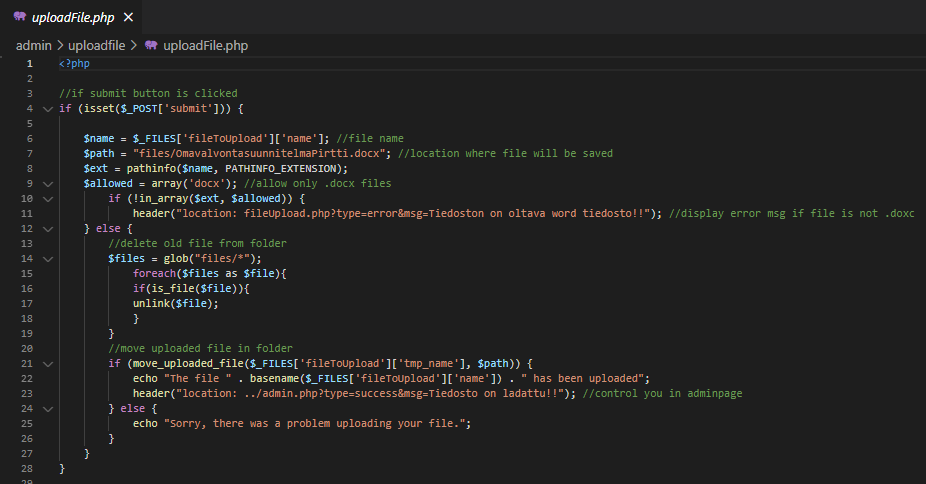


* Session alotetaan.
* Katsotaan kirjautuminen.
* Lisätään bootstrap linkit.
* Tyylittely.



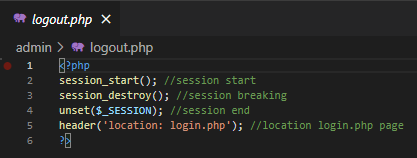
* Takaisin painike tiedonhallintasivulle.
* Error & success ilmotus.
* Formi jossa on action jotenka skripti toimii uploadFile.php kanssa.
* Formiin on lisätty tiedostojen lisääminen multipart/form-data.
* Tiedostoista näytetään vaan word tyyppiset tiedostot accept = ”.docx”.
* Tiedosto tulee näkyviin kansiosta nimellä.

admin/uploadfile/uploadFile.php



* Katsotaan onko submit painiketta painettu.
* $name = katsotaan tiedoston nimi.
* $path = reitti tiedostolle ja laitetaan nimi tiedostolle.
* $ext = pathinfo palauttaa tiedon tallentamis polusta. Sitten laitetaan PATHINFO\_EXTENSION joka näyttää tiedoston tyypin päätteen. esim. Word tiedostossa on .docx pääte.
* Chekataan onko tiedoston pääte docx.
* Kun upataan uutta tiedostoa poistetaan samalla vanha tiedosto pois kansiosta.
* Files/\* poistaa kaikki tiedostot kansiosta.
* unlink avulla tiedosto poistuu kansiosta.
* move\_uploaded\_file siirtää tiedoston. $path avulla siirretään tiedosto oikeaan paikkaan oikealla nimellä.
* Katsotaan tuleeko erroreja.

admin/logout.php

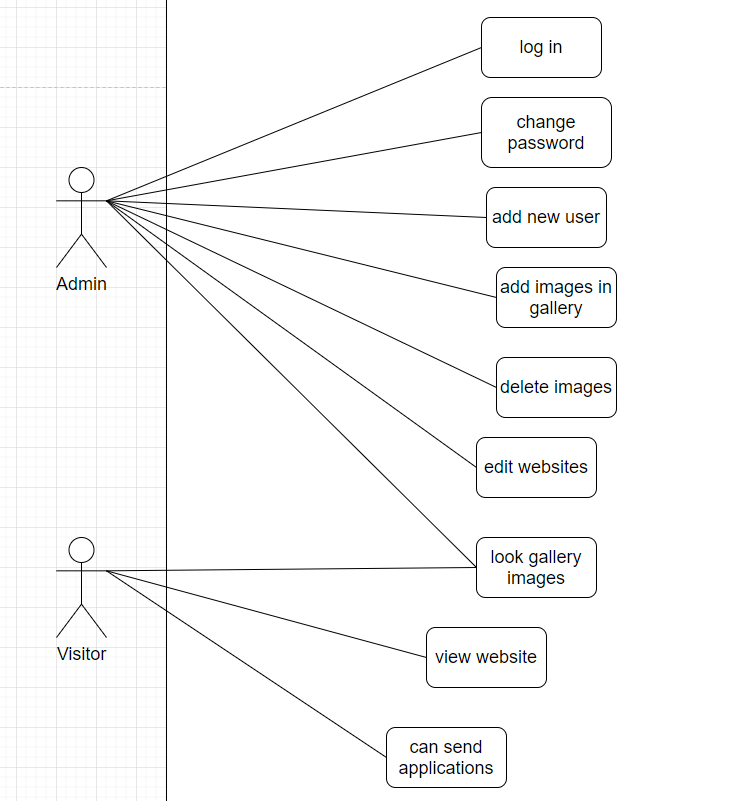


* Session alkaa.
* Session tuhotaan.
* Session loppuu.
* Siirretään login sivulle.

ER-kaavio



UseCase-kaavio



SQL-Database taulut

