TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Henri Tein

Veebilehe loomine firmale Arhitektuurinurk OÜ

Bakalaureusetöö (9 EAP)

Juhendaja:
Vambola Leping

Tartu 2025

Veebilehe loomine firmale Arhitektuurinurk OÜ

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärk on kavandada ja realiseerida kaasaegne, andmebaasipõhine veebileht arhitektuuribüroole Arhitektuurinurk OÜ. Veebileht sisaldab avalehte, kus kuvatakse ettevõtte tutvustus, kontaktandmed ning automaatselt liikuv pildiliugur valminud projektidest. Eraldi galerii vaheleht loodi selleks, et esitada varasemaid projekte üksikasjalikumalt, näidates nii projektide infot kui pilte, mida laetakse dünaamiliselt PostgreSQL andmebaasist. Lisaks töötati välja administraatori liides, mis on kaitstud sisselogimisega ja võimaldab sisuhaldust – näiteks projektide ja slaidide lisamist, muutmist ja kustutamist – ilma eriteadmisi vajamata. Süsteem järgib modulaarset kolme kihi arhitektuuri ning on üles ehitatud HTML-i, CSS-i, JavaScripti, PHP-d ja PostgreSQL-i kasutamisele. Valminud lahendus on mobiilisõbralik, jõudlusele optimeeritud ning seda on testitud erinevates seadmetes ja brauserites. Töö tulemusena loodud veebileht vastab nii funktsionaalsetele kui ka mittefunktsionaalsetele nõuetele.

Võtmesõnad: veebileht, PostgreSQL, galerii, HTML, CSS, Javascript, PHP

CERCS: P175 Informaatika, süsteemiteooria

Creating a website for Arhitektuurinurk OÜ

Abstract:

The goal of this bachelor's thesis is to design and implement a modern, database-driven website for the architecture firm Arhitektuurinurk OÜ. The website consists of a homepage that features an introduction to the firm, contact information, and an automatically rotating image slider of projects. A separate gallery page was created to present the firm's past projects in more detail, including project information and images, all retrieved dynamically from a PostgreSQL database. Additionally, an administrative interface was developed, protected by login authentication, allowing content management such as adding, editing, and deleting projects and slider entries without requiring technical skills. The system follows a modular three-tier architecture and was built using HTML, CSS, JavaScript, PHP, and PostgreSQL. The final solution is responsive, optimized for performance, and tested across devices and browsers, meeting both functional and non-functional requirements.

Keywords: website, PostgreSQL, gallery, HTML, CSS, Javascript, PHP

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Digitaalajastu ettevõtluses on professionaalne ja hästi läbimõeldud veebileht muutunud iga ettevõtte lahutamatuks osaks. See kehtib eriti loovtööstuste, nagu arhitektuuribüroode puhul, kus visuaalne identiteet, tööde esitus ja usaldusväärsus mängivad olulist rolli kliendisuhete loomisel. Just arhitektuuribüroodele on veebileht sageli esmane kontaktikanal, mille kaudu kujuneb esmamulje ettevõtte stiilist, kvaliteedist ja kogemusest.

Arhitektuuribüroo Arhitektuurinurk OÜ vajas uut veebilehte, kuna olemasolev lahendus oli sisuliselt piiratud, visuaalselt ajale jalgu jäänud ning ei võimaldanud projektiportfelli piisaval määral esile tuua. Uue lahenduse vajadus tulenes soovist pakkuda külastajatele selget ja esteetilist ülevaadet ettevõtte tegevusest ning luua võimalus hallata dünaamilist sisu ka tehniliste teadmisteta kasutajatel.

Käesoleva bakalaureusetöö eesmärk on luua andmebaasipõhine veebileht, mis koosneb kahest põhivahelehest – avaleht ja galerii – ning administreerimisliidesest, mille kaudu saab hallata projektide ja liugurite sisu. Leht peab vastama kaasaegsetele kujundus- ja kasutusmugavuse nõuetele ning olema optimeeritud erinevate seadmete ja ekraanisuuruste jaoks.

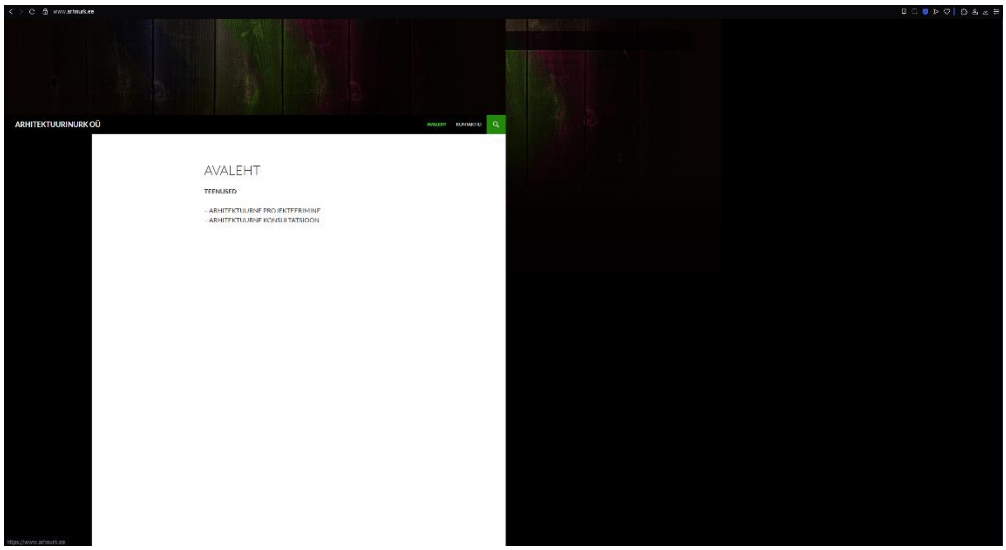
Töö koosneb seitsmest peatükist. Esimeses peatükis antakse ülevaade veebiarenduse ja -disaini taustast ning arhitektuuribüroode veebilehtede ülesehitusest. Teises peatükis määratletakse loodava veebilahenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Kolmandas peatükis kirjeldatakse loodud süsteemi arhitektuuri, andmebaasi struktuuri ja kasutajaliidese elemente. Neljandas peatükis tutvustatakse kasutatud tehnoloogiaid ja arendusvalikuid. Viiendas peatükis kirjeldatakse valminud veebilehte ja selle funktsionaalsust. Kuuendas peatükis käsitletakse loodud lahenduse testimist ja toimivuse hindamist. Viimases peatükis võetakse töö kokku ja antakse ülevaade projekti tulemustest. Lisades on esitatud ekraanipildid veebilehe erinevatest vaadetest, sealhulgas avalehest, galeriist, hüpikaknast (*pop-up window*) ja haldusliidestest (vt Lisad 1–6). Lisadest leiab käesoleva töö raames valminud veebilehe lähtekoodi kasutusjuhendi ja sellele juurdepääsu võimaluse (vt Lisa 7).

1. Veebiarenduse ja -disaini taust

Veebilehe loomisel arhitektuuribüroole tuleb arvestada mitmete erialaspetsiifiliste ootustega, mis puudutavad visuaalset selgust, esteetilist muljet ja projektide esituse kvaliteeti. Käesolev peatükk annab esmalt ülevaate Arhitektuurinurk OÜ varasemast veebilehest ja selle kitsaskohtadest. Seejärel käsitletakse kaasaegse veebidisaini olulisi põhimõtteid, keskendudes visuaalsele hierarhiale, kasutajakogemusele ja reageerivale ülesehitusele. Lõpetuseks antakse ülevaade Eesti arhitektuuribüroode kodulehtedest, mille põhjal kujunes arusaam valdkonnas levinud kujunduslahendustest ja sisu struktuuridest.

1.1 Olemasoleva veebilehe kriitika

Töö alguses oli Arhitektuurinurk OÜ-l olemas kahe vahelehega veebileht, milleks olid avaleht ja kontaktide vaheleht (Joonis 1). Veebilehe kujundus oli väga algeline: see koosnes tumedast taustast, keskele paigutatud tekstiplokist ning lihtsast horisontaalsest navigeerimisribast. Visuaalne esteetika ei toetanud arhitektuuribüroo loomelaadset ja kvaliteedile orienteeritud kuvandit – puudus isikupära, selge kujunduskeel ning sisuline fookus.



Joonis 1. Arhitektuurinurga vana veebileht

Oluline puudujääk oli projektide esitusvõimaluse puudumine – veebilehel ei olnud ühtegi pilti ega visuaalset viidet ettevõtte senisele tööle. Arvestades, et arhitektuur on visuaalselt orienteeritud valdkond, mõjus see veebileht professionaalse portfoolio seisukohast ebasobivana. Külastajal ei olnud võimalik saada ülevaadet ettevõtte stiilist, ulatusest ega kogemusest.

Lisaks oli veebileht staatiline ja mitte interaktiivne – kõik sisu oli fikseeritud ning puudusid dünaamilised komponendid, nagu pildigaleriid, slaidid või kasutajamugavust toetavad funktsioonid. Kuigi leht oli täisekraanil loetav, paiknes kogu sisu ainult lehe vasakul poolel, mis jättis ebaproportsionaalse ja tasakaalustamata mulje ning tekitas tühjuse efekti.

Kujunduselt puudus ka jalus, mis tavaliselt sisaldab korduvat või viidetega infot ning tugevdab struktuurset sidusust.

Kokkuvõttes ei vastanud olemasolev veebileht ei visuaalselt ega funktsionaalselt kaasaegse arhitektuuribüroo vajadustele. Sellest tulenevalt otsustati luua uus veebileht, mis oleks andmebaasipõhine, sisukeskne, visuaalselt esteetiline ning hallatav ka ilma tehniliste teadmisteta. Uue lahenduse eesmärgiks oli usaldusväärse, läbipaistva ja professionaalse kuvandi toetamine ning projektide esiletõstmine visuaalsel kujul.

1.2 Kaasaegse veebilehe olemus

Tänapäevane veebileht on enamat kui lihtsalt staatiline infoleht. See peab olema dünaamiline, mobiilisõbralik ning ligipääsetav, pakkudes kasutajale intuitiivset navigeerimist ja sujuvat visuaalset kogemust [1]. Lisaks sisulisele funktsionaalsusele peab iga veebilehe komponent olema eesmärgipõhine ning toetama kasutaja loomulikku liikumist lehel [2].

Veebilehtede loomisel kasutatakse laialdaselt modulaarset ülesehitust, kus leht jaotatakse loogilisteks plokkideks, nagu päis, navigatsioonimenüü, sisualad, galeriid, hüpikaknad ja jalus. Selline struktuur parandab sisuhalduse paindlikkust ning lihtsustab hilisemat hooldust ja täiendamist [3].

Eriti oluline on reageeriv disain (*responsive design*), mis võimaldab veebilehel kohanduda automaatselt eri seadmete ekraani suurustele. Kasutades näiteks CSS-i meediapäringuid ja paindlikke paigutuslahendusi, saab tagada, et sisu jääb loetavaks ja visuaalselt tasakaalukaks nii mobiilseadmes kui lauaarvutis [4].

1.3 Visuaalne disain ja kasutajakogemus

Arhitektuuribüroo veebilehe puhul on visuaalne disain eriti oluline, kuna see kujundab esmamulje ettevõtte stiilist ja kvaliteedist. Hea veebilehe disain ei tähenda üksnes esteetilist visuaali, vaid eeskätt kasutajasõbralikku ülesehitust, mis aitab kasutajal info kiiresti ja pingutuseta leida [2].

Disaini loomisel lähtuti järgmistest põhimõtetest:

- **Selge visuaalne hierarhia ja ruumipaigutus** – sisu on jaotatud loogilisteks sektsioonideks, mis juhivad kasutaja tähelepanu vastavalt olulisusele (nt projektid, tutvustus, kontaktid) [5].

- **Visuaalne puhtus** – välditi liigset kujunduslikku müra. Kasutati minimalistlikku kujundust, kus pildivalik domineerib teksti üle [6].

- **Responsiivsus** – kõik disainielemendid skaleeruvad vastavalt ekraani suurusele ja asendile. See võimaldab mugavat navigeerimist igas seadmes [4].

- **Tüpograafia ja värvilahendus** – kasutati neutraalset *sans-serif* kirjatüüpi (`Inter`), mis toetab arhitektuuriteemalist sisu. Tausta toonid ja gradientide kasutus jäi tagasihoidlikuks, et mitte konkureerida projekti fotodega [6].

Kokkuvõttes on hea disain see, mis ei sunni kasutajat mõtlema, vaid suunab teda loomulikult – nii visuaalselt kui funktsionaalselt – soovitud tegevuste ja infoni [2].

1.4 Eesti arhitektuuribüroode veebilehtede analüüs

Uue veebilehe loomise eelduseks oli taustauuring Eesti arhitektuuribüroode veebilehtede ülesehituse, disaini ja funktsionaalsuse kohta. Analüüsi käigus analüüsiti järgmiste arhitektuuribüroode kodulehti:

- 3+1 Arhitektid – https://www.threeplusone.ee
- KAOS Arhitektid – https://www.kaosarhitektid.ee
- KOKO Arhitektid – www.koko.ee
- Salto Arhitektid – www.salto.ee
- Kauss Arhitektuur – https://www.kauss.ee
- Arhitektuuribüroo Pluss – www.arhitektuuriburoopluss.ee
- B² Architecture – https://b2architecture.eu
- ATELIER – https://www.atelier.ee

Uurimise fookuses oli lehtede sisuline struktuur, projektide visuaalne esitus, interaktiivsete elementide olemasolu ning kasutajamugavuse lahendused. Leiti, et enamikul juhtudel on arhitektuuribüroode veebilehed kujundatud visuaalset esitust toetavalt, kasutades minimalistlikke kujunduspõhimõtteid. Lehtede keskseks komponendiks on pildigalerii, mida sageli täiendab projekti lühikirjeldus. Pildid on tavaliselt paigutatud ruudustikku ning neile lisatakse hõljumisefekt, mis aitab projekti nime esile tõsta. Klõpsates projektile, avaneb tihti hüpikaken või viiakse külastaja uuele vahelehele, kus kuvatakse detailsem info ja rohkem visuaalset materjali.

Navigatsioon on lihtne ja jagatud loogilisteks jaotisteks, näiteks „Projektid“, „Meist“, „Kontakt“. Värvilahendused on valdavalt neutraalsed – must, valge ja hall toon domineerivad ning erksamaid toone kasutatakse vaid aktsendina. Kirjatüübid on valdavalt *sans-serif*, tagades hea loetavuse ja tänapäevase esteetika. Kasutajaliidesed on reageerivad, kohandudes sujuvalt erinevate seadmete ekraani laiustele, sealhulgas mobiilidele.

Selle analüüsi põhjal selgus, et valdkonnaspetsiifiline veebileht peab keskenduma tugevale visuaalsele kihile, lihtsusele ning kaasaegsele tehnilisele lahendusele. Tulemused andsid sisulise ja esteetilise raamistiku Arhitektuurinurk OÜ uue veebilehe arenduseks, mille keskseks osaks kujunes samuti dünaamiline projekti galerii ja minimaalne kujundus.

2. Projekti nõuded

Käesoleva töö eesmärk on luua arhitektuuribüroo veebileht, mis oleks lihtsasti kasutatav, visuaalselt atraktiivne ning võimaldaks sisuhaldust ilma tehniliste teadmisteta. Selleks sõnastati projekti arenduse alguses järgmised funktsionaalsed ja mittefunktsionaalsed nõuded.

2.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded määratlevad tegevused, mida kasutaja saab veebilehel teha, ja süsteemi käitumise erinevates olukordades [7].

1. **Navigatsioonimenüü on olemas kõigil lehtedel:**
    - Navigatsioonimenüüst saab liikuda nii „Avalehele“ kui ka „Galerii“ vahelehele sõltumata aktiivsest vahelehest.
    - Navigatsioonimenüül olev logo suunab alati tagasi avalehele.
    - Navigatsioonimenüü on fikseeritud (*sticky*) ja jääb nähtavaks ka lehe kerimisel.

2. **Avalehel on bänner-sektsioon koos taustapildi ja tekstiga:**
    - Bänner-sektsioon katab lehe ülaosa ja sisaldab arhitektuuribüroo visuaalset tutvustust.
    - Taustapildile kuvatakse ettevõtte nimi või lühike hüüdlause.
    - Bänner-sektsiooni all kuvatakse kontaktinfo ja visuaalne eristus bänneri ning sisu vahel.

3. **Esilehel kuvatakse dünaamiline pildi liugur:**
    - Liuguri andmed laetakse andmebaasist.
    - Liugur kuvab iga slaidi kohta pildi, pealkirja ja infovälja.
    - Liugur liigub automaatselt, sisaldab navigeerimisnooli ja punktindikaatoreid.

4. **Esilehel kuvatakse ettevõtte kontaktandmed jaluses:**
    - Jaluses on nähtaval ettevõtte e-posti aadress ja telefoninumber.
    - Jalus kuvatakse igal veebilehe alalehel.

5. **Galerii vaheleht kuvab kõik projektid ülevaatlikult:**
    - Projektide andmed laetakse andmebaasist.

- Iga projekti juures kuvatakse nimi ja esinduspilt.
- Projektid paigutatakse *responsive grid*-tüüpi paigutusse

6. **Projektide detailvaade avaneb hüpikaknas:**
    - Klikkides projektile, laetakse detailsemad andmed andmebaasist projekti kohta.
    - Hüpikaknas kuvatakse projekti nimi, suurus ruutmeetrites, valmimisaasta ja kirjeldus.
    - Projekti pildid kuvatakse Swiper.js põhisel pildigalerii liuguril hüpikakna sees.
    - Hüpikaknas on olemas sulgemisnupp ja slaidide vahetamine noolte või punktindikaatorite abil.

7. **Administraatori vaade on kaitstud ja varjatud tavakasutaja eest:**
    - Admin-paneelile pääseb ainult otselinkide kaudu, see pole navigeerimismenüüs nähtav.
    - Admin-paneel on kaitstud kasutaja autentimisega.

8. **Administraatori vaates on võimalik hallata projektide andmeid:**
    - Administraator saab lisada uusi projekte.
    - Administraator saab muuta olemasolevate projektide andmeid.
    - Administraator saab kustutada projekte.
    - Administraator saab iga projekti juurde lisada pilte.
    - Administraator saab pilte projektide küljest eemaldada.
    - Administraator saab muuta projekti esipilti.

1. **Administraatori vaates on võimalik hallata liuguri sisu:**
    - Administraator saab lisada uusi slaide.
    - Administraator saab muuta olemasolevate slaidide infot ja järjekorda.
    - Administraator saab eemaldada slaide.

2.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsed nõuded kirjeldavad süsteemi omadusi, mis ei puuduta otseselt funktsionaalsust, vaid pigem kvaliteeti, jõudlust, laiendatavust ja kasutajakogemust [7].

1. **Veebileht peab olema mobiilisõbralik (*responsive design*):**
    - Paigutus peab kohanema automaatselt ekraani suurusega.
    - Galerii, liugur ja hüpikaken peavad töötama nii telefonis kui arvutis.

2. **Veebileht peab olema kiiresti laaditav:**
    - Lehe algne laadimisaeg peab jääma alla 3 sekundi normaalse ühenduse korral.
    - Lehel kasutatavad pildid peavad olema optimeeritud ning andmete laadimine peab toimuma asünkroonselt, et vähendada alglaadimise koormust.

3. **Veebileht peab olema kasutajasõbralik:**
    - Struktuur peab olema intuitiivne ja arusaadav.
    - Hüpikakna avamine/sulgemine ja liikumine galeriis peab olema sujuv.

4. **Veebileht peab olema turvaline:**
    - Sisestatud andmed valideeritakse serveri poolel.
    - Admin-funktsioonid ei ole avalikult nähtavad ega ligipääsetavad ilma admin-kasutajasse sisse logimata.

5. **Veebileht peab olema ühilduv levinumate brauseritega:**
    - Chrome, Firefox, Safari, Edge
    - Veebileht peab säilitama funktsionaalsuse ja disaini kõigis toetatud brauserites.

3. Arhitektuur

Veebilehe arhitektuur on kavandatud selliselt, et see oleks loogiliselt üles ehitatud, modulaarselt laiendatav ning toetaks andmebaasipõhist sisuhaldust. Süsteemi ülesehitus järgib klassikalist kliendi-serveri mudelit, kus kasutajaliides suhtleb serveri ja andmebaasiga läbi kindlate andmevoogude. Arhitektuuri eesmärk on tagada selgus, lihtsus ja paindlikkus nii kasutaja kui ka administraatori vaates [8, 9].

3.1 Üldine süsteemimudel

Veebirakendus koosneb kolmest peamisest kihist:

- **Kasutajaliidese kiht (*frontend*)** – sisaldab HTML-, CSS- ja JavaScript-komponente, mis loovad veebilehe visuaali ning võimaldavad kasutajaga suhelda [1, 2].

- **Serveripoolne kiht (*backend*)** – töötleb kasutaja päringuid, suhtleb andmebaasiga ja tagastab vajaliku sisu JSON-formaadis [10].

- **Andmebaasi kiht** – talletab kogu dünaamilise sisu: projektide info, pildid, slaidid [10, 4].

Andmete liikumine toimub järgmiselt: kasutajaliides saadab päringu, server töötleb selle ja küsib andmed andmebaasist, tagastades tulemuse kasutajale JSON-formaadis. Selline kihiline arhitektuur võimaldab süsteemi lihtsalt laiendada ja hooldada [8].

3.2 Kasutajaliidese struktuur

Kasutajale kuvatav leht koosneb mitmest sisulisest sektsioonist.

- **Navigatsioonimenüü**: fikseeritult ülemises servas, võimaldab liikuda kahe põhilehe vahel.

- **Avaleht**: sisaldab bänner-sektsiooni, dünaamilist liugurit ja kontaktinfot.

- **Galeriivaade**: kuvatakse kõik projektid visuaalsel kujul koos hõljumisefektide ja detailidega.

- **Hüpikaken projektide detailide jaoks**: avaneb pildile klõpsamisel ja kuvab projekti detailid ning lisapildid.

- **Jalus**: sisaldab kontaktandmeid ja lehe autorsuse infot.

Kogu liides on kujundatud *responsive*-disaini põhimõtteid järgides, et tagada kasutatavus erinevates seadmetes [4, 10].

3.3 Serveripoolne loogika

Serveri ülesandeks on vahendada andmevahetust kasutaja ja andmebaasi vahel.

- Selleks saadetakse projekti andmed galeriivaatesse, kus need renderdatakse kliendi poolel.

- Päringu kaudu saadakse üksiku projekti detailandmed, mida kasutatakse hüpikakna vaate täitmiseks.

- Liuguri sisu saadetakse eraldi failist (PHP kaudu), mis genereerib JSON-andmed ja edastab need JavaScriptile.

- Administraatori tegevused, nagu sisestamine, kustutamine ja muutmine, valideeritakse serveris ning salvestatakse andmebaasi.

Kõik päringud läbivad sisendi valideerimise ja järgivad turvalise andmetöötluse põhimõtteid [8].

3.4 Andmebaasi struktuur

Andmebaas on loodud relatsioonilise mudeli järgi. Andmestruktuur sisaldab järgmisi komponente:

- **projektide tabel**
  – sisaldab iga projekti peamist informatsiooni: nimi, valmimisaasta, ruutmeetrid ja kirjeldus.
  – igal projektil on unikaalne identifikaator, mis võimaldab seoste loomist piltidega

- **piltide tabel**
  – sisaldab pildi failinime ja viidet sellele, millise projektiga see seotud on

– võimaldab siduda ühe projektiga mitu pilti, kuid mitte ühte pilti mitme projektiga

- **liuguri sisu tabel**
  – sisaldab iga slaidi kohta pealkirja, info välja ja pildi asukoha
  – võimaldab hallata avalehel kuvatava pildi liugurit

Andmebaasi ülesehitus võimaldab hõlpsalt lisada, muuta või kustutada sisu ilma HTML- või CSS-faile muutmata [4].

3.5 Administreerimisliidese arhitektuur

Administreerimisliides on eraldiseisev osa veebilehest, mis on tavakasutajale nähtamatu. See võimaldab

- lisada uusi projekte, sisestades kõik seotud andmed ja pildid;
- muuta olemasolevate projektide andmeid;
- kustutada projekte;
- lisada ja hallata liugureid, mis kuvatakse avalehel;
- hallata projektiga seotud pilte eraldi.

Admin-liidese toimingud on jagatud eraldi vaadeteks, kus iga tegevus keskendub konkreetsele sisutüübile (nt projektid, slaidid) [8].

4. Kasutatud tehnoloogiad

Veebilehe arendamisel kasutati mitmeid tehnoloogiaid, mille eesmärk oli luua kaasaegne, andmebaasipõhine ja kasutajasõbralik süsteem. Käesolevas peatükis antakse esmalt ülevaade kasutatud põhitehnoloogiatest – HTML, CSS ja JavaScript – koos selgitustega nende rollist veebiarenduses. Seejärel tutvustatakse serveripoolset loogikat, andmebaasi kasutamist ning kolmandate osapoolte teeke, mis aitasid lisada interaktiivsust ja parandada kasutajakogemust. Lisaks käsitletakse failistruktuuri ülesehitust ja võrreldakse valitud lahendusi võimalike alternatiivsete tehnoloogiatega, et põhjendada tehtud valikuid projekti kontekstis.

4.1 HTML

HTML (*HyperText Markup Language*) on veebiarenduse aluskeel, mille abil määratakse kindlaks veebilehe struktuur ja sisu loogiline jaotus [11]. Iga veebileht koosneb HTML-elementidest, mis tähistavad erinevaid osi, nagu pealkirjad, lõigud, lingid, pildid või nupud [11]. HTML ei määra, milline sisu visuaalselt välja näeb, vaid loob struktuurse aluse, millele saab hiljem rakendada kujundust ja funktsionaalsust [12].

Käesolevas projektis kasutati **HTML5** versiooni, mis toob kaasa mitmeid uuendusi, sealhulgas semantilised elemendid, nagu *<header>*, *<nav>*, *<section>*, *<article>* ja *<footer>* [11]. Need aitavad otsingumootoritel ja abistavatel tehnoloogiatel veebilehe sisust paremini aru saada ja parandavad ligipääsetavust [12]. HTML võimaldas luua kogu veebilehe skeleti, sealhulgas navigeerimisriba, sisualad, galeriiplokid ja hüpikaknad [11]. HTML5 võimaldab struktureerida sisu selgemini ja parandada nii kasutajakogemust kui ka otsingutulemeid, eriti kui seda kombineerida kaasaegsete kujundustehnikatega [11, 12].

Projektis kasutati HTML-i, et luua kõik veebilehe vaated alates avalehest kuni galeriivaate ja haldusliideseni. Kõik olulised sektsioonid, nagu päis, bänner, slaider ja galeriiplokk on HTML-i kaudu struktureeritud loogilisteks plokkideks, mis toetavad nii visuaalset kui funktsionaalset kihistust.

4.2 CSS

CSS (*Cascading Style Sheets*) on keelevorm, mille abil kujundatakse HTML-is määratud sisu [11]. CSS määrab, kuidas elemendid veebilehel välja näevad – sealhulgas värvid, fondid, paigutus, vahed, efektid ja vastavus eri ekraani suurustele [12].

Projektis kasutati **CSS3** standardit koos kaasaegsete paigutusmeetoditega, nagu *Flexbox* ja *Grid*, mis võimaldavad elemente täpselt joondada ja organiseerida [12]. Kujundus loodi käsitsi, ilma valmisraamistikke (nt Bootstrap või Tailwind) kasutamata, et saavutada puhas, minimalistlik ja täpse kontrolliga lahendus, mis sobib hästi arhitektuuri valdkonda [12].

Lisaks rakendati *media queries'ete* abil reageerivat disaini – see tähendab, et veebilehe paigutus kohandub vastavalt seadme ekraani suurusele, pakkudes mugavat kasutuskogemust nii telefonis, tahvelarvutis kui arvutis [13]. CSS3 koos reageeriva disaini praktikatega on üks

olulisemaid eeldusi tänapäevaste veebirakenduste esteetika ja funktsionaalsuse ühendamisel [12, 13].

Projektis loodi kogu kujundus käsitsi, kasutades CSS-i, et määratleda värvid, paigutus, fondid, vahed ja kujundusdetailid. Lisaks loodi galeriivaates hõljumisefektid, mis lisab veebilehele interaktiivsust.

4.3 JavaScript

JavaScript on veebilehtede interaktiivsuse lisamiseks kasutatav programmeerimiskeel [4]. Kui HTML loob struktuuri ja CSS kujundab välimuse, siis JavaScript võimaldab dünaamilisi tegevusi – näiteks nuppudele vajutamisega seotud toiminguid, andmete laadimist serverist ilma lehte uuesti avamata või animatsioonide ja liugurite loomist [12].

Projektis rakendati JavaScripti eelkõige kolmes võtmefunktsioonis:

- **pildiliuguri haldamine** – slaidid laaditakse serverist ja kuvatakse automaatselt liikuvas galeriis;
- **galerii projektide kuvamine** – JavaScripti abil loetakse andmed andmebaasist ning kuvatakse need kasutajale ruudustikuna;
- **hüpikaknad** – projektide detailne info (nt kirjeldused ja mitu pilti) avanevad eraldi vaates, kui kasutaja klõpsab pildile.

Andmevahetuseks kasutati funktsiooni *fetch()*, mis võimaldab asünkroonset suhtlust serveriga ilma lehte uuesti laadimata [12]. Kasutajaliidest hallati DOM API kaudu, mis võimaldab dünaamiliselt luua, eemaldada ja muuta HTML-elemente [13]. JavaScript võimaldab luua kaasaegseid, kiireid ja kasutajasõbralikke veebilehti, kus sisu reageerib kasutaja tegevusele reaalajas [4, 12].

Projektis oli JavaScript keskne tööriist dünaamilise sisu laadimiseks serverist – näiteks pildigalerii, projektide info ja slaidid kuvati *fetch()*-i abil reaalajas. Lisaks võimaldas JavaScript hüpikakende avamist ja sulgemist, piltide vahetust galeriis ja automaatse pildiliuguri funktsionaalsust avalehel.

4.4 Serveripoolne loogika

Serveripoolne loogika tegeleb veebilehe taustal toimuvate protsessidega, mis ei ole kasutajale otseselt nähtavad, kuid on olulised andmete töötlemiseks ja haldamiseks. Selles projektis kasutati serveripoolse loogika teostamiseks PHP-d, mis on laialdaselt kasutatav avatud

lähtekoodiga serveripoolne skriptimiskeel. PHP võimaldab luua dünaamilisi veebilehti, suhelda andmebaasidega ja töödelda vormides sisestatud andmeid[14].

PHP skriptid käivitatakse serveris, mis tähendab, et kasutaja brauser saab ainult lõpliku HTML-i, ilma et näeks PHP koodi. See võimaldab paremat turvalisust ja kontrolli andmete üle [14].

Käesolevas projektis kasutati PHP-d andmebaasi päringute tegemiseks, vormide valideerimiseks ja andmete töötlemiseks. Näiteks kui kasutaja lisab uue projekti, saadetakse andmed PHP skriptile, mis valideerib sisendi ja salvestab info PostgreSQL andmebaasi. Samamoodi haldavad PHP-failid ka liuguri muutmist ja sisu kustutamist administraatori vaates.

4.5 PostgreSQL

PostgreSQL on võimas, avatud lähtekoodiga relatsiooniline andmebaasi haldur, mis on tuntud oma töökindluse, standardite toega ja keerukate päringute käsitlemise võimekuse poolest [15]. PostgreSQL toetab rikkalikke andmetüüpe, andmetervikluse mehhanisme (nt võõrvõtmed) ning keerulisi transaktsioone, mis muudab selle sobivaks nii väiksematele kui suurematele rakendustele [15].

Projektis kasutati PostgreSQL-i kolme peamise tabeli kaudu:

- **projektid** – salvestab iga projekti kohta pealkirja, valmimisaasta, ruutmeetrid ja kirjelduse;
- **pildid** – salvestab iga projektiga seotud galerii piltide failinimed ning seob need vastava projektiga võõrvõtme kaudu;
- **slider_pildid** – sisaldab avalehel kasutatavate slaidide andmeid, sealhulgas pildi linke, pealkirju ja visuaalseid märgendeid.

Andmed kuvatakse lõppkasutajale JavaScripti kaudu, kuid kogu sisu pärineb dünaamiliselt PostgreSQL-ist, mida teenindavad PHP-põhised API-failid.

4.6 Kolmandate osapoolte teegid

Veebiarenduses kasutatakse sageli kolmandate osapoolte teeke, et lisada lehele funktsionaalsust ilma, et arendaja peaks seda nullist looma. Antud projektis kasutati peamiselt Swiper.js teeki.

Swiper.js on kaasaegne JavaScripti teek, mis võimaldab luua sujuvaid ja mobiilseid liugureid, toetades nii automaatset kui käsitsi navigeerimist [16]. Siinses projektis kuvati selle abil avalehel pildiliugurit, kus iga slaid esindas ühte projekti ja galerii vahelehel asuvat galeriid.

Swiper võimaldas visuaalselt sujuvaid üleminekuid ning puutetundlikku juhtimist väiksemates seadmetes. See teek aitas saavutada professionaalse ja tänapäevase kasutajakogemuse.

4.7 Failistruktuuri ülesehitus

Selge ja loogiline failistruktuur on veebiprojekti hooldatavuse ja arenduse seisukohalt hädavajalik. Antud projektis järgiti modulaarset ülesehitust, jagades funktsioonid eraldi kaustadesse:

- **admin/** – sisaldab administraatori vaate PHP-faile (sisestamine, muutmine, kustutamine, vaadete haldus);
- **scripts/** – JavaScripti failid galeriide ja liugurite juhtimiseks;
- **style/** – sisaldab kõik CSS-failid;
- **assets/** – sisaldab kõiki pilte ja fontide faile.

Selline jagamine võimaldab teistel arendajatel projektis kiiresti orienteeruda ning muudab ka hilisema täiendamise lihtsamaks. Failistruktuuri loogika on kooskõlas kaasaegsete veebirakenduste soovitustega [2, 17].

4.8 Alternatiivsed tehnoloogiad ja tehtud valikute põhjendus

Veebilehe loomisel kaaluti mitmeid alternatiivseid tehnoloogilisi lahendusi, kuid lõplikud valikud lähtusid projekti iseloomust, eesmärgist ja ressursikasutusest. Eesmärk ei olnud luua üleliia keerukat süsteemi, vaid säilitada kontroll koodi ja kujunduse üle ning saavutada funktsionaalne, esteetiline ja laiendatav tulemus.

Üks võimalus oleks olnud kasutada sisuhaldussüsteeme, nagu WordPress või Drupal, mis võimaldavad hallata sisu ilma koodi kirjutamata. Sellised platvormid on sobivad, kui eesmärgiks on luua kiiresti standardne veebileht koos valmis funktsioonidega, kuid nende kasutamine toob sageli kaasa piiranguid kujundusvabaduses ning kaasneb mittevajalikke funktsionaalsusi, mis võivad koormata väiksemahulist ja kohandatud lahendust [13]. Käsitsi arendus annab arendajale täieliku kontrolli nii struktuuri, kujunduse kui funktsionaalsuse üle ning väldib CMS-platvormidele omast keerukust ja sõltuvust [4, 13].

Serveripoolse arenduse puhul kaaluti ka Node.js (JavaScript põhinev serveriraamistik) ja Django (Pythonil põhinev raamistik) kasutamist. Kuigi mõlemad on võimsad ja tänapäeval laialdaselt kasutusel, otsustati nende asemel kasutada PHP-d. Põhjuseks oli PHP lihtne

seadistamine, lai levik ning sobivus väikese või keskmise mahuga projektidele, mis ei vaja eraldi API-kihti [4, 14].

Kujunduse poolelt jäeti kõrvale CSS raamistikud, nagu Bootstrap ja Tailwind CSS, kuna need eeldavad klasside intensiivset kasutust ja võivad piirata kujunduse täpset kontrolli. Selle asemel loodi kujundus käsitsi, kasutades CSS3 paigutustehnikaid (`Flexbox` ja `Grid`), mis võimaldasid saavutada täpselt soovitud tulemuse ilma üleliigsete stiilikihtideta [12].

Alternatiivina oleks võinud kaaluda ka suuremaid JavaScripti raamistikke, nagu React, Vue või Angular. Nende kasutamine oleks andnud võimaluse rakendada komponentpõhist arendust ja SPA-lähenemist (*single-page application*). Selline lahendus ei olnud aga käesoleva projekti ulatuse ja eesmärkide puhul põhjendatud. Selle asemel valiti kerged ja konkreetse funktsiooniga JS-teegid, nagu Swiper.js, mis sobitus hästi käsitsi loodud HTML-struktuuriga ja võimaldas säilitada parema kontrolli koodi üle [16].

Lisaks kasutati iseseisvalt loodud failistruktuuri, mitte mõne suurema raamistikuga kaasnevat jäika struktuuri. See võimaldas loogilist ja projekti vajadustest lähtuvalt kaustajaotust ning lihtsustas hilisemat arendust ja haldust [17, 18].

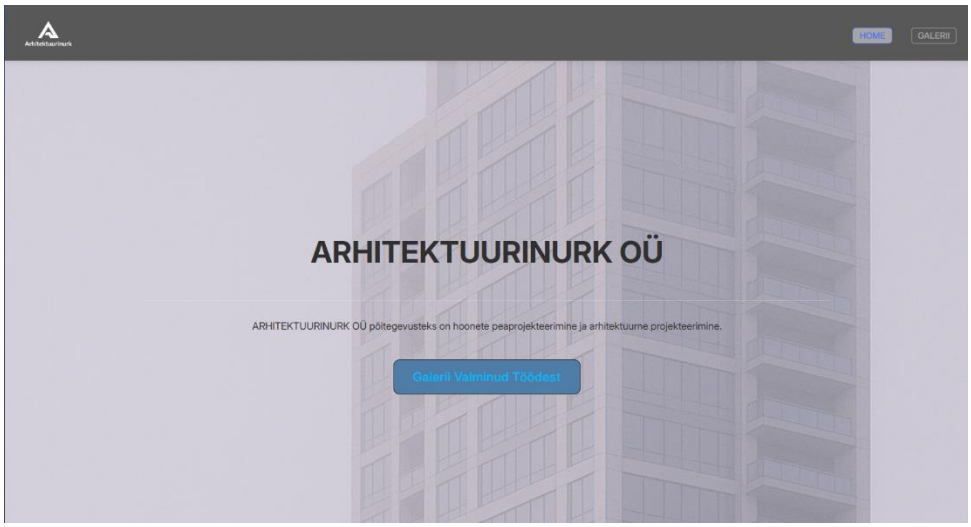
5. Tulemus

Süsteemi arenduse tulemusena valmis arhitektuuribüroo veebileht, mis koosneb kahest põhivahelehest: avaleht ja galerii. Lisaks loodi haldusliides, mis võimaldab sisuhaldust ilma otsese koodimuudatuseta. Valminud veebileht vastab nii funktsionaalsetele kui ka mittefunktsionaalsetele nõuetele, pakkudes kasutajale mugavat ja visuaalselt terviklikku kogemust.

5.1 Avaleht

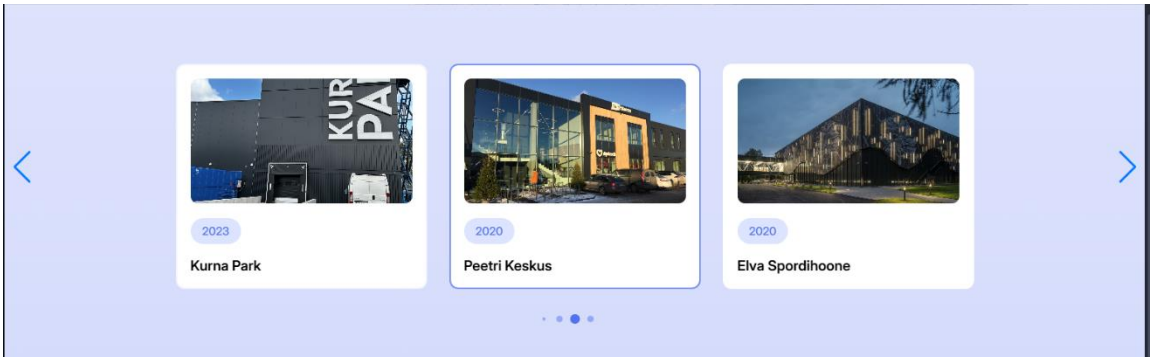
Avaleht annab esmamulje veebilehest ning tutvustab arhitektuuribüroo identiteeti. See koosneb kolmest põhikomponendist:

1. Bänner-sektsioon – visuaalselt silmapaistev ala, mille taustal on arhitektuuriteemaline pilt ning peal tutvustav tekst (Joonis 2).



Joonis 2. Bänneri ala avalehel

2. Pildiliugur – dünaamiline visuaalne element, mis kuvab järjest erinevaid pilte koos pealkirjade ja märksõnadega. Liugur liigub automaatselt ning võimaldab ka manuaalset navigeerimist nooltega või punktindikaatoritega (Joonis 3).



Joonis 3. Pildiliugur avalehel

3. Kontaktinfoplokk ja jalus – lehe allosas kuvatakse ettevõtte põhiandmed: e-post, telefoninumber ning autoriõiguste märge (Joonis 4).

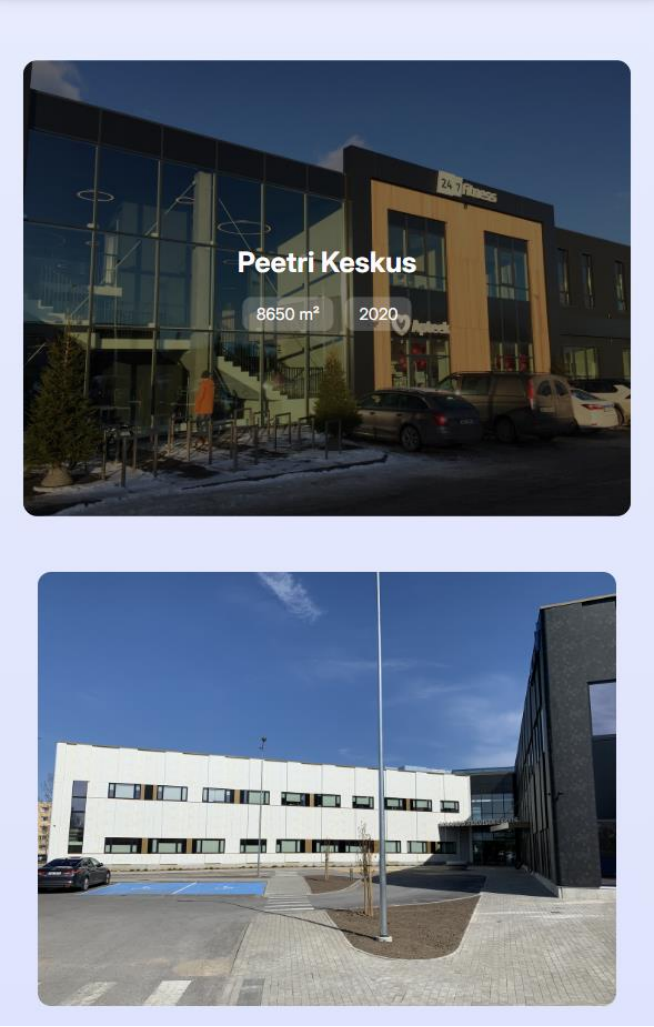
Joonis 4. Kontaktinfo avalehel

Avalehe eesmärk on luua professionaalne ja visuaalselt haarav esmamulje, suunata kasutajat lehel edasi liikuma ning esitada arhitektuuri esteetikat läbi piltide.

5.2 Galeriivaade

Galerii vaheleht võimaldab sirvida kõiki lisatud projekte ning avada igaühe kohta detailse info.

1. **Projektide ülevaade** – projektid kuvatakse ruudustikuna, kus igaüks sisaldab pealkirja ja esinduspilt. Iga pildi kohal aktiveerub hõljumisefekt, mis muudab pildi tumedamaks ja kuvab kesksele alale projekti nime, ruutmeetrid ja valmimisaasta (Joonis 5).

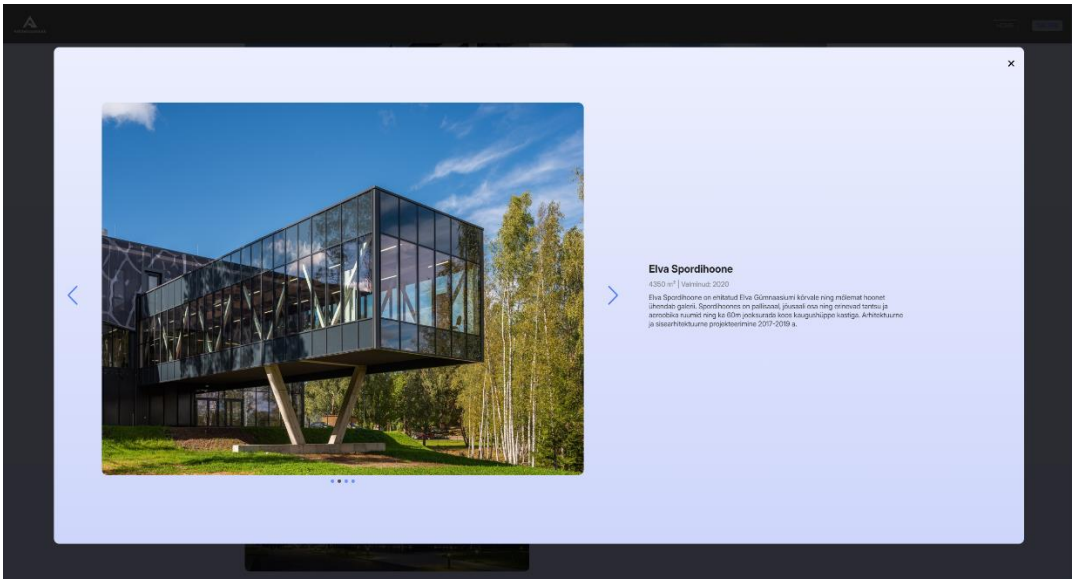


Joonis 5. Pildigalerii galerii vahelehel, kus on ühe pildi peale liigutud hiirekursoriga

2. **Interaktiivne avamine** – klikkides projektile, avaneb hüpikaken, mis kuvab

    - projekti pealkirja, suuruse ruutmeetrites, valmimisaasta ja kirjelduse;
    - pildigalerii, kus on võimalik liikuda projektiga seotud piltide vahel;

3. **Hüpikaken** – sulgemiseks on eraldi nupp, slaidide liikumine toimub noole klikkidega ning üleminekud on visuaalselt sujuvad (Joonis 6).



Joonis 6. Hüpikakna vaade projektist

Galerii on keskne osa veebilehest, kuna see annab külastajale visuaalse ülevaate ettevõtte tööst ja kvaliteedist.

5.3 Haldusliides

Administreerimisliides loodi selleks, et veebilehe sisu oleks võimalik hallata otse brauserist, ilma et oleks vaja muuta lähtekoodi või andmebaasi käsitsi. Liidese eesmärk on muuta sisuhaldus võimalikult lihtsaks ja visuaalselt arusaadavaks ka mittetehnilisele kasutajale.

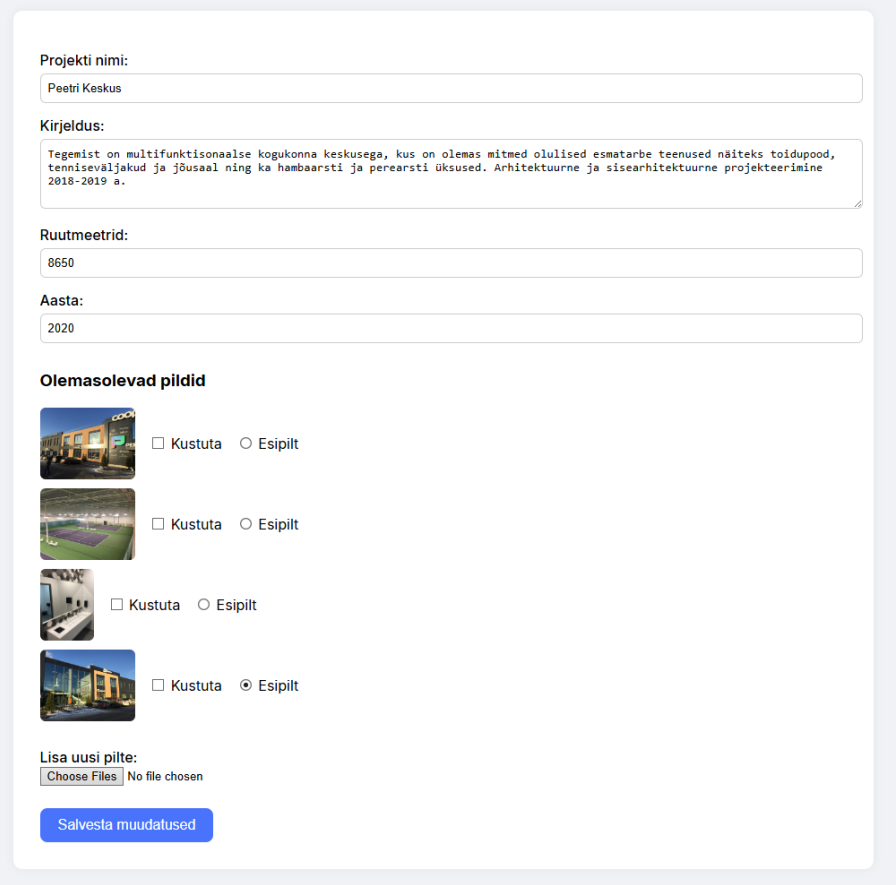
Haldusliides on eraldatud tavakasutaja vaadetest ning sellele pääseb ligi ainult otselingi kaudu. Selle kujundus järgib lihtsat ja loogilist ülesehitust, kus iga sisutüüp (projektid ja liuguri sisu) on jagatud eraldi vaadeteks.

Projektide haldamise vaade

Projektide halduseks on loodud eraldi vaade, kus kuvatakse andmebaasis olevate projektide tabelvaade. Iga kirje juures on võimalik

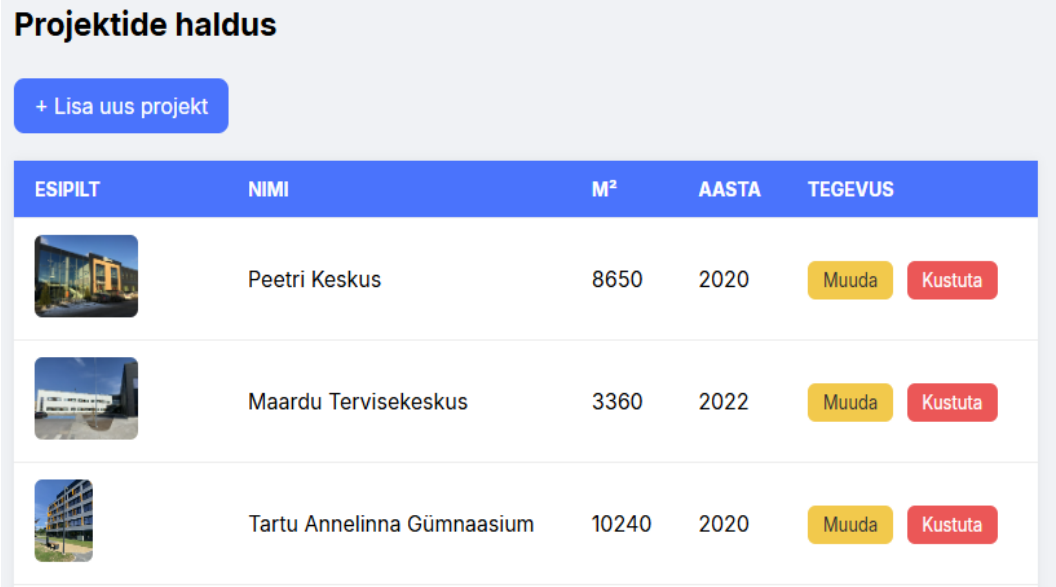
- **redigeerida projekti andmeid**, sealhulgas nime, valmimisaastat, suurust ja kirjeldust;

- **hallata projektiga seotud pilte**, kus on võimalik lisada või eemaldada fotosid (Joonis 7);



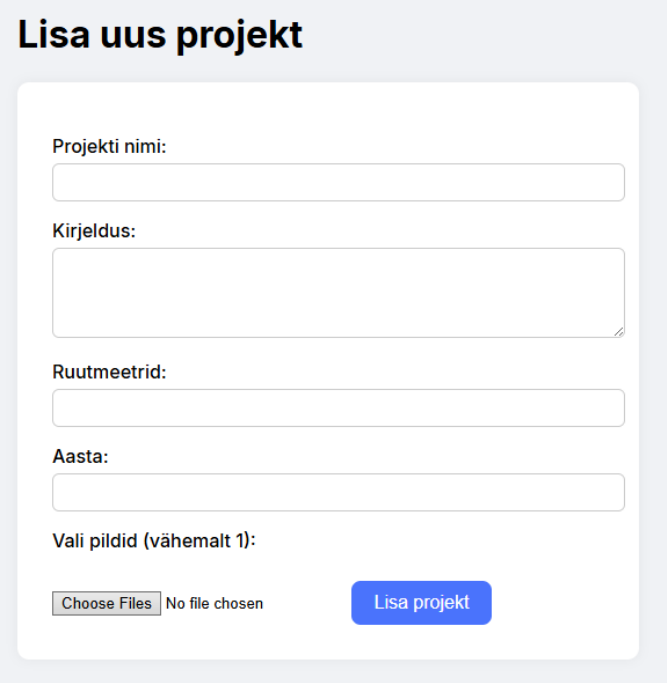
Joonis 7. Projekti info, piltide ja esipildi muutmise vaade

- **kustutada projekt**, mille tulemusel eemaldatakse seotud andmed ka andmebaasist ja kustutatakse pildid kaustast (Joonis 8).



Joonis 8. Projektide haldusvaade, kus saab muuta ja kustutada projekte

Uue projekti lisamiseks on olemas eraldi vorm, mis sisaldab kõiki vajalikke välju (Joonis 9). Projekti salvestamisel tehakse serverile päring, mis sisestab andmed andmebaasi ja seob üleslaetud pildid vastava projektiga.



Joonis 9. Projekti lisamise vaade

Failide üleslaadimine toimub vormi kaudu, serveripoolne skript teisaldab need kindlasse kausta ja lisab info andmebaasi koos projekti ID-ga.

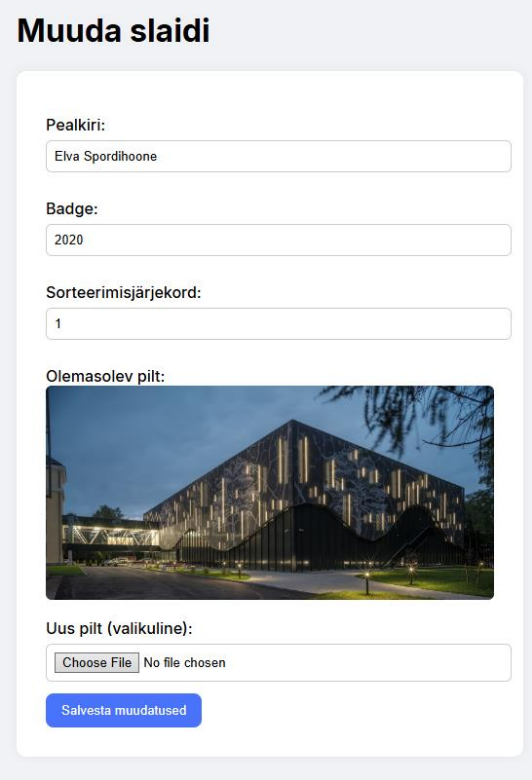
Liuguri sisu haldamise vaade

Liuguri vaade võimaldab muuta avalehel kuvatavat dünaamilist pildiliugurit. Iga slaid sisaldab

- pealkirja;
- infovälja;
- kujunduspilti.

Administraator saab lisada uue slaidi, muuta olemasolevaid, muuta nende järjekorda või need kustutada (Joonis 10). Uue slaidi lisamisel kontrollitakse vormis kõik väljad ja server valideerib sisendi enne andmebaasi salvestamist.

Liuguri muudatused kajastuvad koheselt ka avalehel, kuna leht laeb selle sisu reaalajas serverilt.

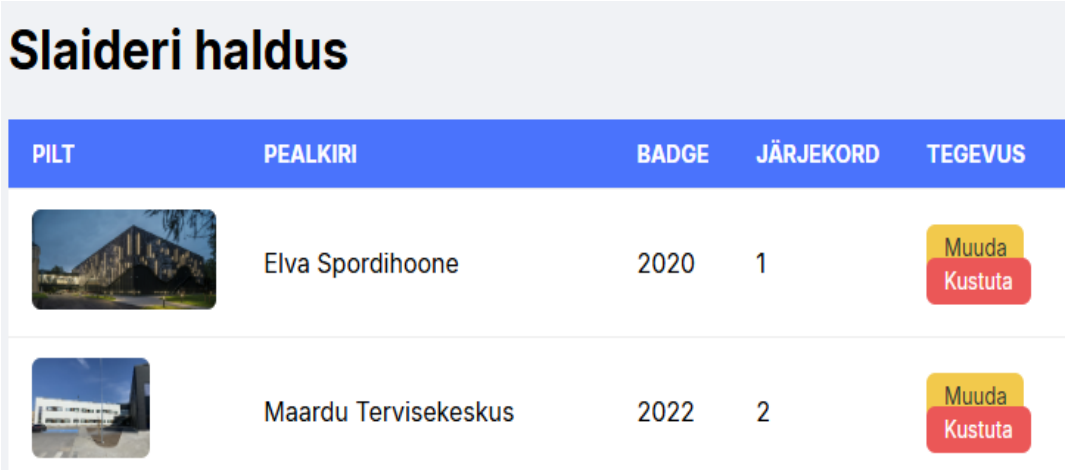


Joonis 10. Liuguri ühe slaidi muutmise vaade

Liidese tööloogika ja andmevoog

Haldusliidese iga toiming on seotud konkreetse serveripoolse funktsiooniga, mis

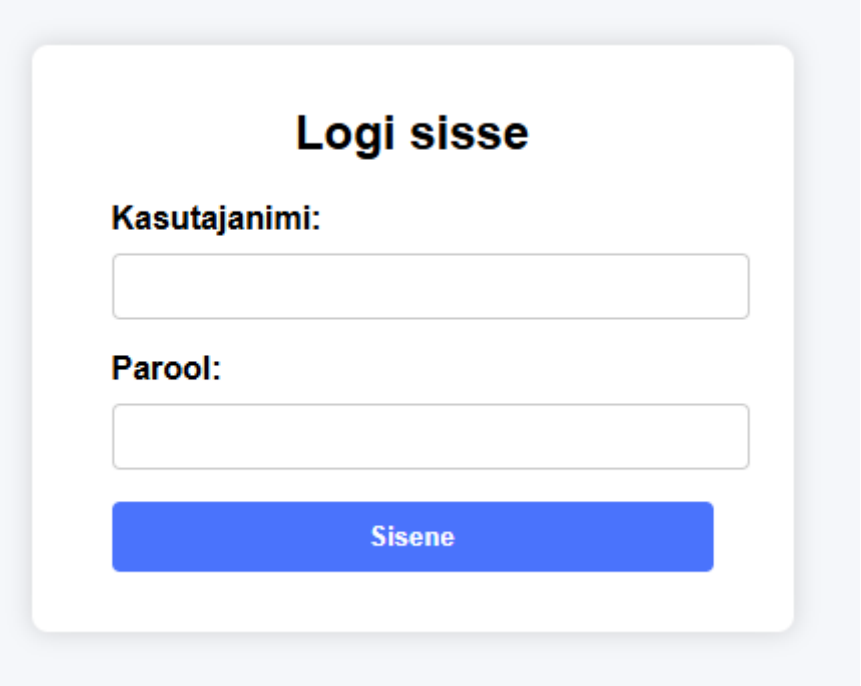
1. kontrollib, kas edastatud andmed on korrektse kujuga;
2. teostab vastava andmebaasi päringu (INSERT, UPDATE, DELETE);
3. tagastab kasutajaliidesele kinnituse õnnestumise või veateate kohta.



Joonis 11. Vaade liugurihaldusest, kus saab muuta ja kustutada liuguri slaide

Visuaalselt jaguneb haldusliides kaheks põhikategooriaks: projektid ja slaidid. Mõlemal juhul toimub andmete muutmine sünkroonis andmebaasiga – seetõttu iga sisestus või muudatus salvestatakse kohe ning see mõjutab kasutajaliideses kuvatavat sisu.

Ligipääs administreerimisliidesele on kaitstud sisselogimisvaatega, kus kasutaja peab sisestama kasutajanime ja parooli (Joonis 12). Ligipääsu taotlust kontrollib server ning administraatori vaated ei ole ilma autentimiseta kättesaadavad. See tagab, et sisu muutmise ja kustutamise õigused on ainult volitatud kasutajatel ning andmebaasis olev info on kaitstud juhusliku või pahatahtliku ligipääsu eest. Turvalisuse seisukohalt on serveripoolne autentimine ja andmete valideerimine üks olulisemaid standardeid veebirakenduse arhitektuuris [19].



Joonis 12. Sisselogimise vaade

6. Testimine ja hindamine

Pärast veebilehe valmimist viidi läbi mitmeid testimisetappe, et kontrollida selle funktsionaalsust, visuaalset korrektsust, kasutajakogemust ning tehnilist jõudlust. Testimise eesmärk oli tuvastada võimalikud vead, hinnata süsteemi töökindlust erinevates olukordades ning tagada, et kõik arendatud funktsioonid töötaksid ootuspäraselt [20].

6.1 Funktsionaalne testimine

Funktsionaalne testimine keskendus sellele, kas iga nõutud funktsioon töötab kavandatud viisil. Testiti kõikide avalikke ja haldusvaateid.

Testitud toimingud kasutaja vaates:

- lehtede vahel liikumine navigeerimismenüü kaudu
- bänner-sektsiooni korrektne kuvamine ja reageerimine erinevatele ekraanisuurustele
- pildiliuguri automaatne ja manuaalne liikumine
- galeriis olevate projektide laadimine andmebaasist
- projektide avamine hüpikaknas ja pildigalerii toimimine selles

Testitud toimingud haldusvaates:

- uue projekti lisamine koos mitme pildi üleslaadimisega
- olemasoleva projekti andmete muutmine
- projektide kustutamine ning selle mõju galeriivaatele
- liuguri sisu lisamine, muutmine ja eemaldamine

Kõik funktsioonid töötasid vastavalt ootustele. Projektide ja slaidide andmete muutmised kajastusid koheselt kasutaja vaates, ilma et oleks vaja lehte käsitsi uuendada. Funktsionaalse testimise eesmärk on kontrollida, kas süsteemi iga osa töötab vastavalt spetsifikatsioonile [21].

6.2 Responsiivsuse testimine

Veebilehte testiti erinevates seadmetes ja ekraani suurustes:

- **lauaarvuti (1920×1080):** kõik elemendid paigutusid loogiliselt; liugur ja galerii toimisid sujuvalt;

- **tahvelarvuti (768×1024):** navigeerimismenüü muutus kompaktsemaks, liugur ja hüpikaknad kohandused ekraani suurusele;

- **nutitelefon (360x640):** kogu sisu oli loetav, slaidide vahetamine ja hüpikaknad toimisid sujuvalt, kujunduselemente kuvati kompaktselt.

Veebileht kohandus kõigis testitud seadmetes hästi tänu paindlikule paigutusele ja meedia päringutele [20, 22].

6.3 Brauserite ühilduvuse testimine

Testiti järgmistes brauserites:

- Google Chrome

- Mozilla Firefox

- Microsoft Edge

- Safari (mac OS ja iOS seadmes)

Kõigis brauserites säilis lehe funktsionaalsus ja kujundus. Ainsad väiksemad erinevused esinesid vormistuse detailides (nt fontide renderdamine), kuid need ei mõjutanud kasutatavust. Brauseri testimine on oluline, et tagada töökindlus erinevates keskkondades [23].

Responsiivsuse ja brauserite ühilduvuse testimine viidi läbi BrowserStacki keskkonnas, mis võimaldab simuleerida erinevaid seadmeid ja brauserite versioone ilma füüsiliste seadmeteta [23].

6.4 Jõudlus ja laadimiskiirus

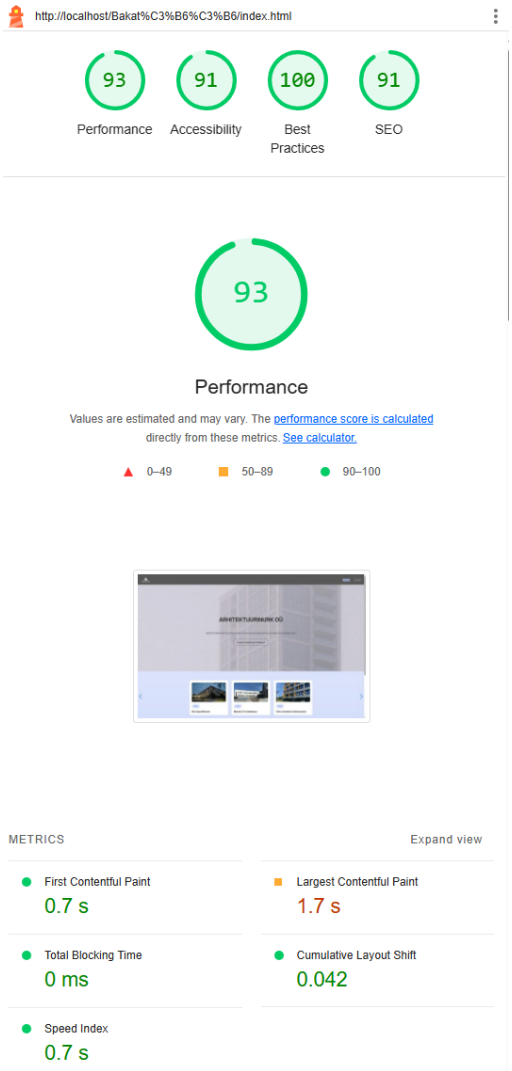
Lehe jõudlust hinnati ka subjektiivselt ning lisaks kasutati arenduse käigus brauserisisest tööriista Lighthouse (Joonised 13 & 14). Tulemused näitasid:

- **kiire alglaadimine:** lehe esmane sisu kuvatakse alla 2 sekundiga (stabiilse ühenduse korral);

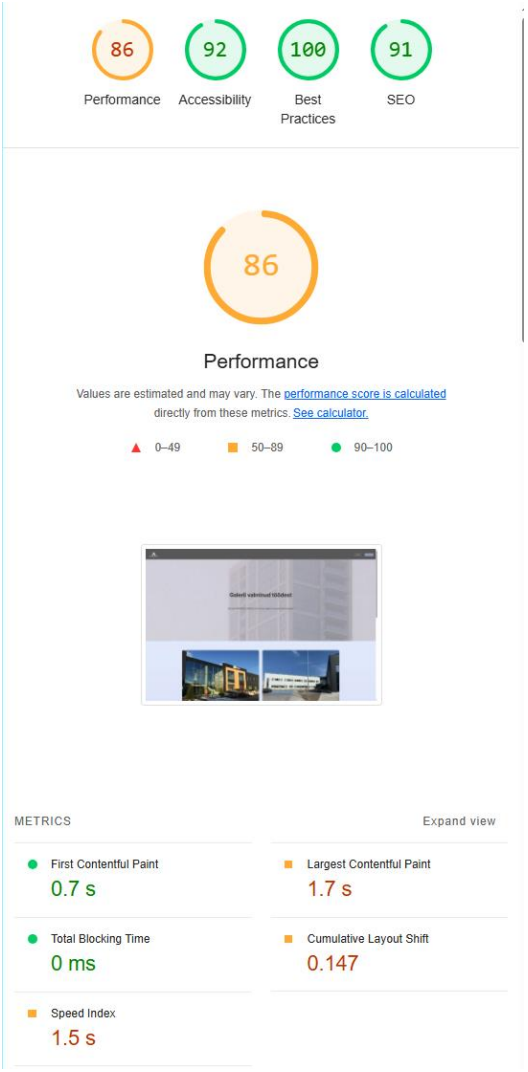
- **optimeeritud pildid:** failide suurused on kontrolli all, mis aitab kaasa jõudlusele;

- **asünkroonne andmelaadimine:** tänu JavaScriptile laetakse sisu vajaduspõhiselt, mis vähendab algkoormust ja kiirendab kasutuskogemust [24].

Süsteemi koormustestimist ulatuslikult ei tehtud, kuid tavakasutuse tingimustes püsis lehe töö stabiilsena.



Joonis 13. Avalehe testimine tööriista Lighthouse abil



Joonis 14. Galerii vahelehe testimine tööriista Lighthouse abil

7. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua arhitektuuribüroole Arhitektuurinurk OÜ andmebaasipõhine, visuaalselt atraktiivne ja lihtsalt hallatav veebileht. Uus lahendus pidi asendama varasema staatilise veebilehe, mis ei sisaldanud pilte, oli raskesti laiendatav ning ei võimaldanud külastajal tutvuda ettevõtte loodud projektidega.

Veebilehe arendamiseks kasutati järgmisi tehnoloogiaid: HTML, CSS ja JavaScript kliendipoolseks loogikaks; PHP serveripoolseks töötlemiseks ning PostgreSQL andmete salvestamiseks. Kujundus loodi käsitsi ilma valmisraamistikke kasutamata, et säilitada täpne kontroll paigutuse ja stiili üle ning sobitada disain arhitektuuri valdkonna ootustega. JavaScripti abil loodi dünaamiline sisu laadimine ja moodulid, mis võimaldasid kasutajatel sujuvalt sirvida projekte ning administraatoril hallata veebilehe sisu ilma tehniliste teadmisteta.

Valminud lahendus sisaldab kahte põhivaadet – avalehte ja galeriid – ning eraldiseisvat haldusliidest, millele pääseb ligi ainult autenditud kasutaja. Avalehel kuvatakse tutvustav bänner, kontaktinfo ja andmebaasist laaditav automaatne pildiliugur. Galeriivaates saab tutvuda ettevõtte loodud projektidega ruudustikuna paigutatud piltide ja detailse hüpikakna kaudu. Administraatori liideses on võimalik projekte lisada, muuta, kustutada ning hallata seotud pilte ja slaide.

Töö käigus valmis kaasaegne veebileht, mis vastab kasutusmugavuse, responsiivsuse ja andmepõhisuse nõuetele. Valminud lahendust testiti erinevates brauserites ja seadmetes, mille tulemused kinnitasid selle stabiilsust ja sobivust praktiliseks kasutuseks. Töö tulemusena loodi terviklik veebiplatvorm, mis toetab Arhitektuurinurk OÜ professionaalset kuvandit ja lihtsustab sisuhaldust tulevikus.

Viidatud kirjandus

1. Lewis JR, Sauro J. USABILITY AND USER EXPERIENCE: DESIGN AND EVALUATION. *ResearchGate,* 2021, https://www.researchgate.net/publication/373487143_USABILITY_AND_USER_EXPERIENCE_DESIGN_AND_EVALUATION (10.05.2025)

2. Batatina B. FIRST CODING STEPS: WEB DESIGN WITH HTML, CSS, AND JAVASCRIPT. *ResearchGate,* 2024, https://www.researchgate.net/publication/379832521_FIRST_CODING_STEPS_WEB_DESIGN_WITH_HTML_CSS_AND_JAVASCRIPT (10.05.2025)

3. Kalkman B. What are HTML, CSS, and PHP? *Rocketmedia* https://rocketmedia.com/resources/what-are-html-css-and-php (10.05.2025)

4. Holst C. Website Usability Best Practices (Backed By Research). *Baymard* https://baymard.com/learn/website-usability (10.05.2025)

5. Knight C, Glaser J. Creating Exciting And Unusual Visual Hierarchies. *Smashing Magazine,* 2013, https://www.smashingmagazine.com/2013/02/creating-visual-hierarchies-typography/ (10.05.2025)

6. Falode E. Color Theory in UI Design: How Colors Affect User Experience. *Cyberogism*, 2024, https://cyberogism.com/color-theory-ui-design/ (10.05.2025)

7. Dave D, Anu V. Identifying Functional and Non-functional Software Requirements From User App Reviews. *IEEE,* 2022, https://ieeexplore.ieee.org/abstract/document/9795770 (10.05.2025)

8. Web Application Architecture: The Basics. *Intellectsoft*, 2024 https://www.intellectsoft.net/blog/web-application-architecture/ (10.05.2025)

9. Andersson M. The Importance of Web Application Architecture. 2023 https://www.diva-portal.org/smash/get/diva2:1772594/ATTACHMENT01.pdf (10.05.2025)

10. Sotnik S, Manakov V, Lyashenko V. Overview: PHP and MySQL Features for Creating Modern Web Projects. *IJAISR*, 2023, https://openarchive.nure.ua/server/api/core/bitstreams/99c2a8e9-aefd-47d4-a06f-838650b760db/content (10.05.2025)

11. Crudu A. The Importance of HTML, CSS, and Javascript in Web Development. *MoldStud*, 2024, https://moldstud.com/articles/p-the-importance-of-html-css-and-javascript-in-web-development (10.05.2025)

12. Mohd TK, Thompson J. Comparative Analysis on Various CSS and JavaScript Frameworks. *ResearchGate*, 2022, https://www.researchgate.net/publication/366079444_Comparative_Analysis_on_Various_CSS_and_JavaScript_Frameworks (10.05.2025)

13. Dhariwal S. CMS Vs Custom Website Development: Picking the Right One *AddWeb. Solution*, 2024, https://www.addwebsolution.com/blog/cms-vs-custom-website-development (10.05.2025)

14. PHP and HTML Forms: Processing Data on the Server Side. *Web crafting code*, https://webcraftingcode.com/html-fundamentals/php-and-html-forms-processing-data-on-the-server-side/ (10.05.2025)

15. Salunke SV, Ouda A. A Performance Benchmark for the PostgreSQL and MySQL Databases. *MDPI*, 2024, https://www.mdpi.com/1999-5903/16/10/382 (10.05.2025)

16. Swiper API. https://swiperjs.com/swiper-api (10.05.2025)

17. Sullivan BO. Folder Structure and Frameworks: What is exerting control? *DEV Community*, 2019, https://dev.to/barryosull/folder-structure-and-frameworks-what-is-exerting-control-4kpi (10.05.2025)

18. Thumati PRR. Modern Web Design: Utilizing HTML5, CSS3, and Responsive Techniques. *Jnrid*, 2020, https://tijer.org/jnrid/viewpaperforall.php?paper=JNRID2308001 (10.05.2025)

19. Secure web application architecture: tips to prevent threats. *Adamo Software*, 2023, https://adamosoft.com/blog/website-development/secure-web-application-architecture/ (10.05.2025)

20. Young D. Software Testing Overview. *ResearchGate*, 2022, https://www.researchgate.net/publication/360852127_Software_Testing_Overview (10.05.2025)

21. Functional and Nonfunctional Requirements Specification. *AltexSoft*, 2023, https://www.altexsoft.com/blog/functional-and-non-functional-requirements-specification-and-types/ (10.05.2025)

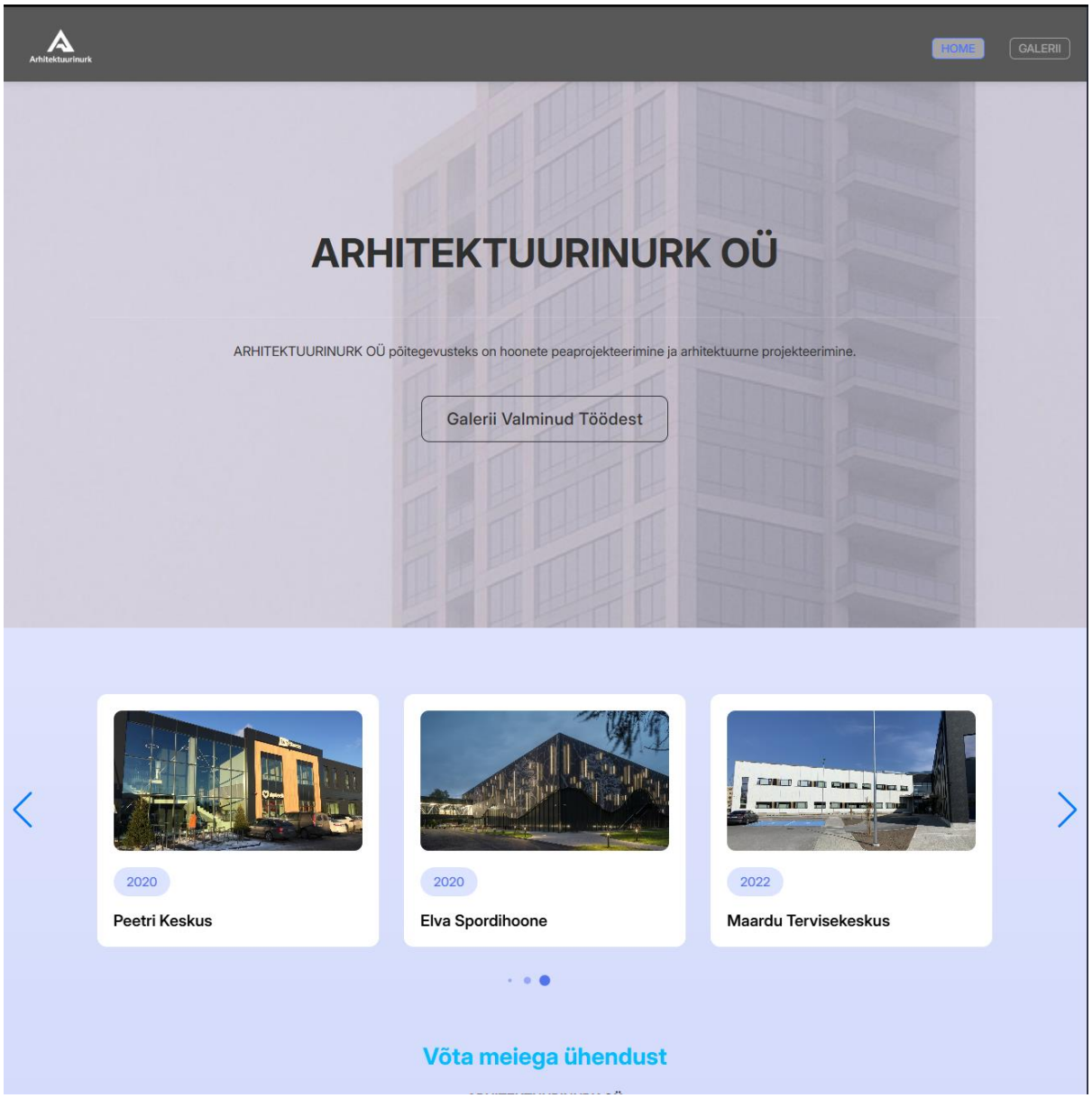
22. Shukla S, Rajan AT, Aravind S, Gupta RK. The Role of HTML5 and CSS3 in Creating Optimized Graphic Prototype Websites and Application Interfaces. *ResearchGate*, 2022, https://www.researchgate.net/publication/387238322_The_Role_of_HTML5_and_CSS3_in_Creating_Optimized_Graphic_Prototype_Websites_and_Application_Interfaces (10.05.2025)

23. Cross Browser Testing: Definition, Why it is Important, How to do it. *BrowserStack*, https://www.browserstack.com/cross-browser-testing (10.05.2025)

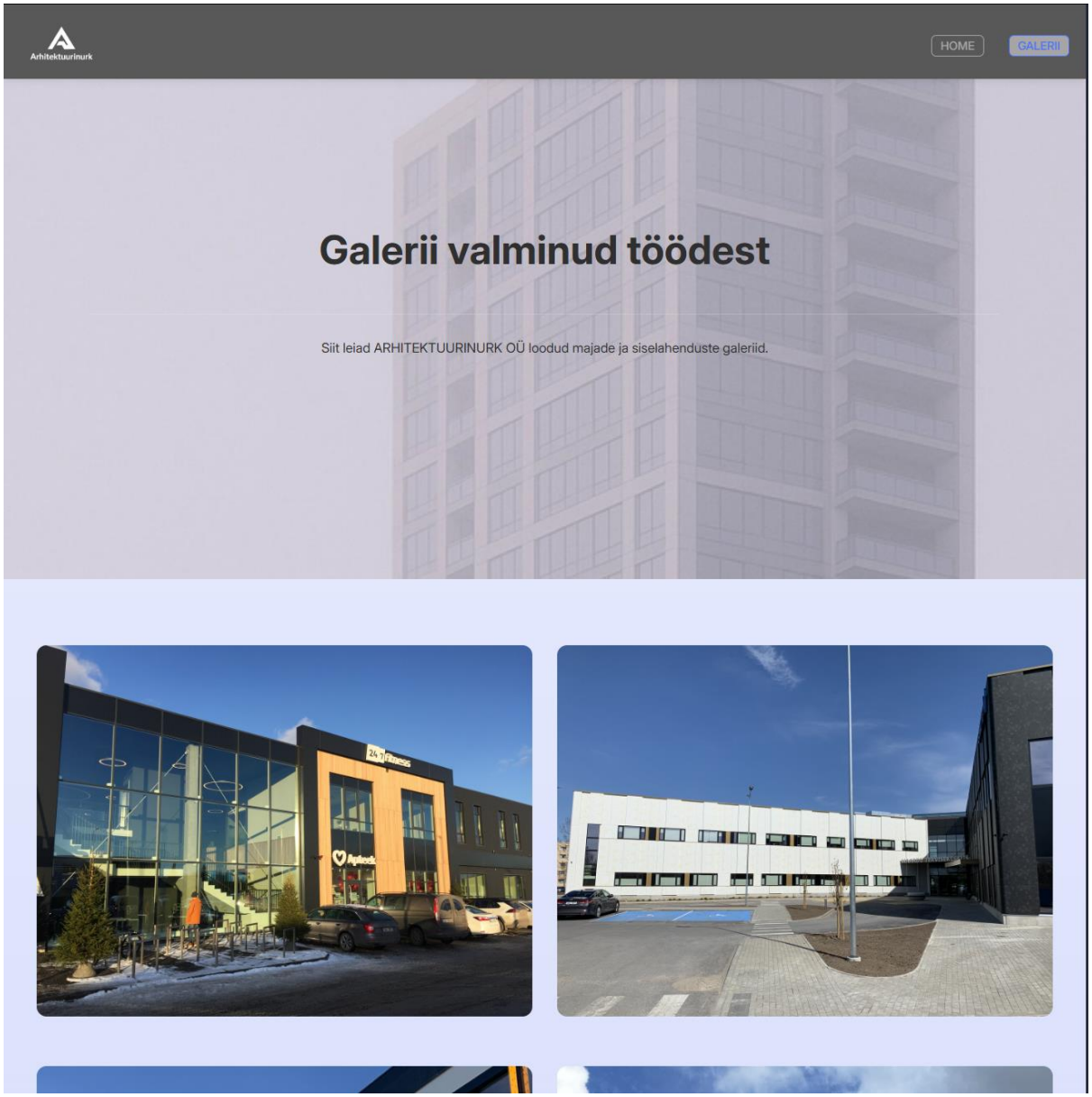
24. Lighthouse performance scoring. *Chrome for Developers*, 2019, https://developer.chrome.com/docs/lighthouse/performance/performance-scoring (10.05.2025)

Lisad

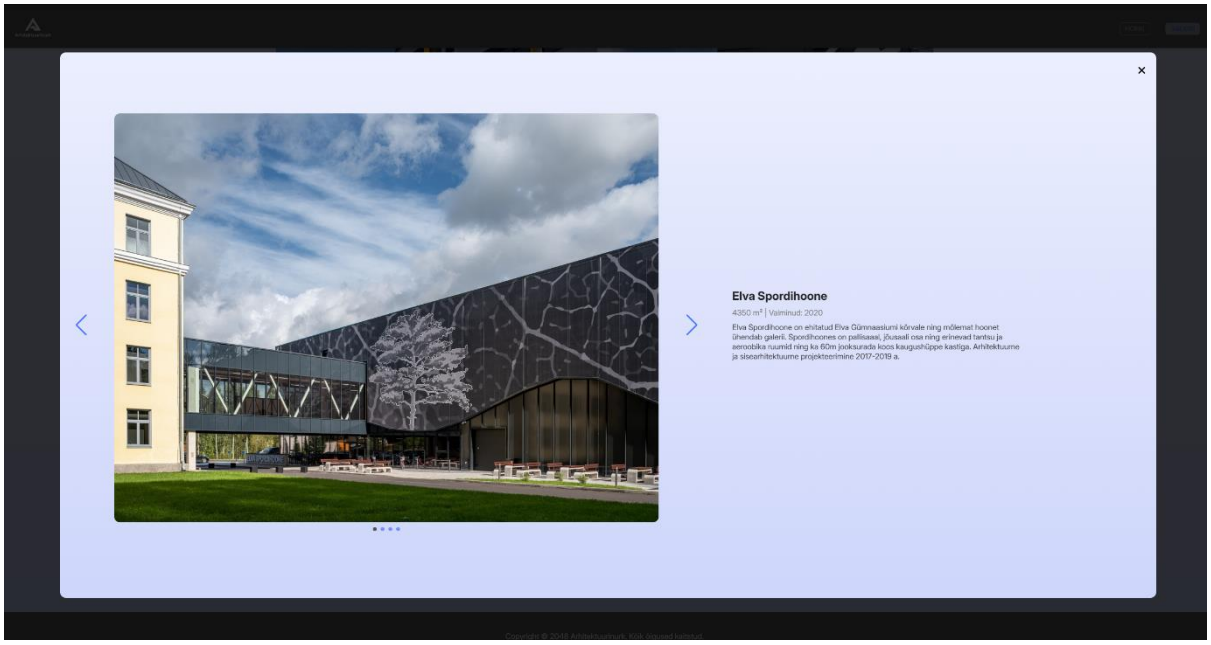
Lisa 1. Avalehe vaade täisekraanil



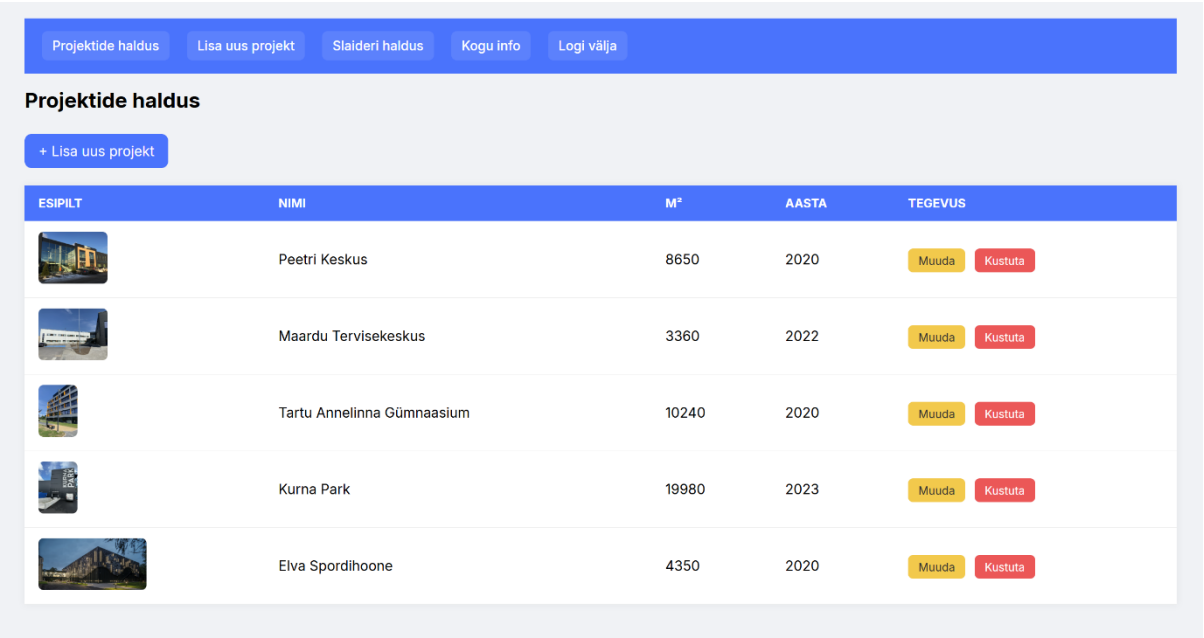
Lisa 2. Galeriivaade täisekraanil



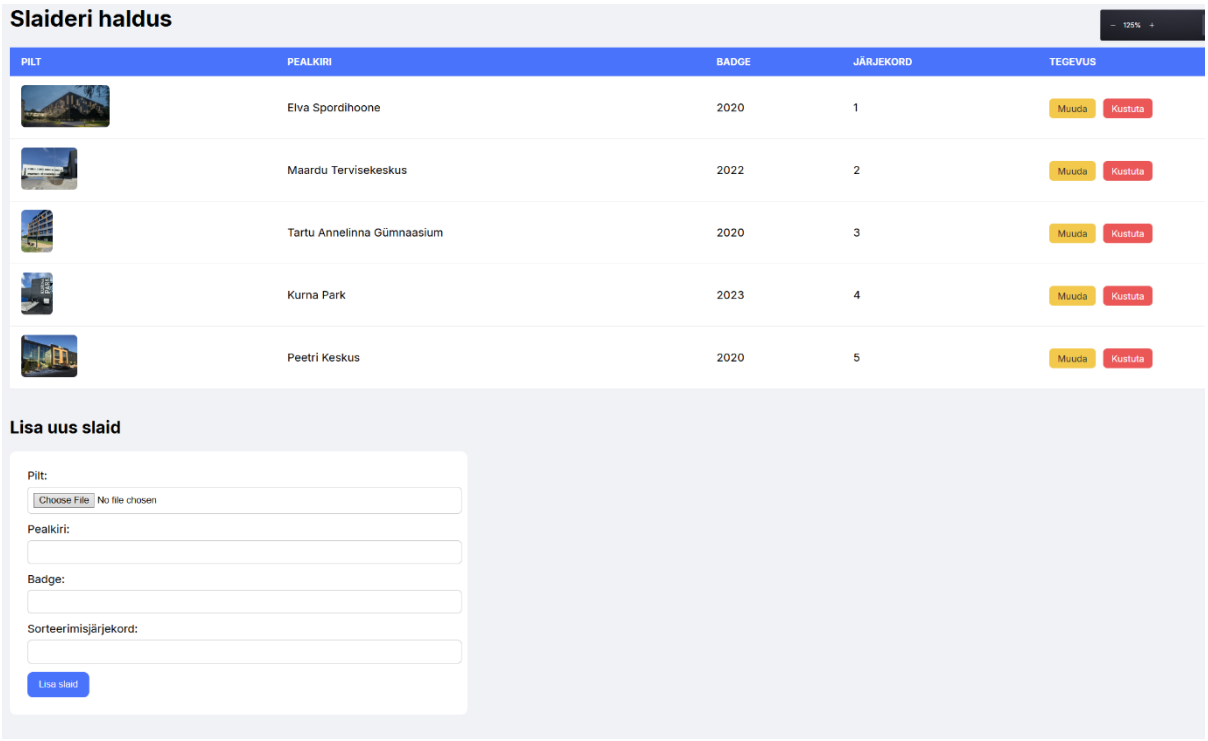
Lisa 3. Hüpikakna vaade täisekraanil



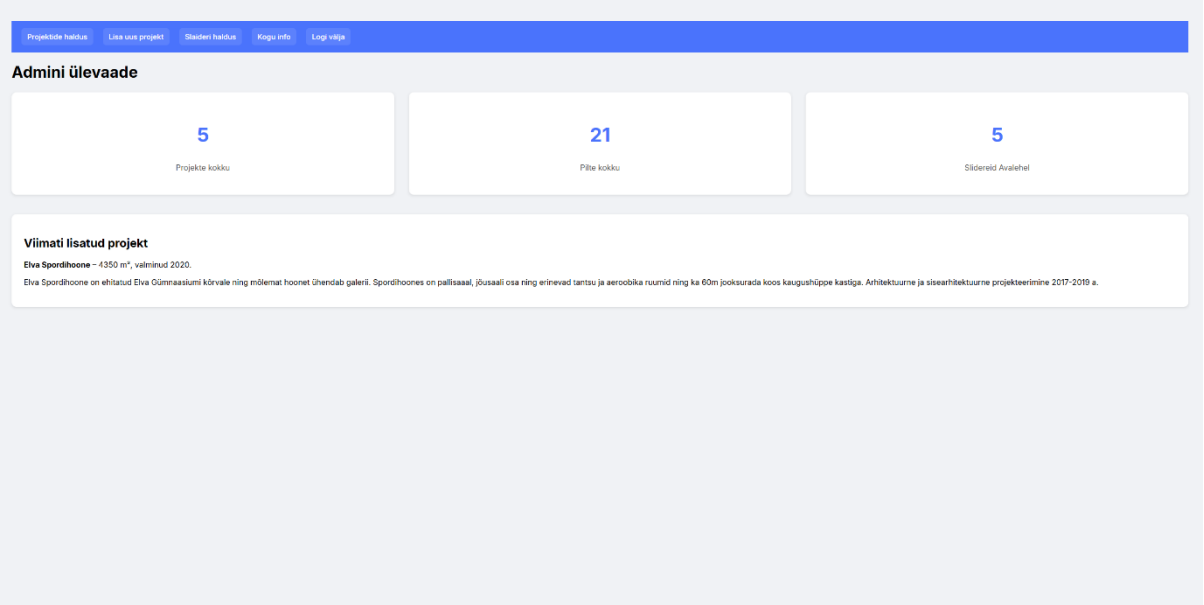
Lisa 4. Projektide haldusvaade täisekraanil



Lisa 5. Liuguri haldusvaade täisekraanil



Lisa 6. Haldusliideses kogu info vaade täisekraanil



Lisa 7. Veebilehe installeerimis- ja kasutusjuhend

Käesoleva töö failid on kättesaadavad järgnevast Google Drive lingist: https://drive.google.com/drive/folders/1MppfrTZE7wPUZeZLb0bzeJ57xCmCNIBM?usp=sharing

Vajalikud programmid veebilehe kasutuseks:

1. Veebiserver (nt XAMPP)
2. PHP (v 7.4 või uuem)
3. PostgreSQL (andmebaasi mootor)
4. PgAdmin (valikuline, PostgreSQL kasutajaliides)
5. Veebibrowser (nt Google Chrome, Firefox, Opera GX jne)

Failide paigutamine:

1. Lae alla kogu projekt Google Drivest.
2. Kui on ZIP failina laetud, siis paki failid lahti.
3. Aseta kogu kaust oma veebiserveri htdocs (XAMPP) kausta, näiteks: C:\xampp\htdocs\Projekt

Andmebaasi Seadistamine:

1. Käivita pgAdmin.
2. Vasakul paneelil klõpsa Databases > paremklõps > Create > Database
3. Pane nimeks arhnurk ja vajuta Save.
4. Ava käsurida (Command Prompt) ja liigu PostgreSQL bin kausta (vajadusel):

```
cd "C:\Program Files\PostgreSQL\17\bin"
```

Kui PostgreSQL on paigaltatud teise versiooniga või teise kausta, siis tuleb kohandada seda käsku vastavalt oma süsteemi seadistusele.

5. Käivita järgmine käsk, et importida andmebaasi sisu:

```
psql -U postgres -d arhnurk -f
"C:\Users\KASUTAJA\Desktop\arhnurk_backup.sql"
```

(asenda tee failini vastavalt oma arhnurk_backup.sql asukohale)

6. Kui küsitakse parooli, sisesta PostgreSQL kasutaja postgres parool.
7. Veendu, et arhnurk andmebaasis on olemas tabelid projektid, pildid, slider_pildid ja need sisaldavad andmeid. Selleks saab pgAdmin rakenduses *query tool'*i kasutades sisestada järgneva koodi:

```
SELECT tablename
FROM pg_tables
WHERE schemaname = 'public';
```

See näitab näiteks:

```
pildid
projektid
slider_pildid
```

Kui tabelite nimed kuvatakse, tähendab see, et import õnnestus. Kui mitte, tuleb kontrollida, kas failis arhnurk_backup.sql on sisu olemas ja kas import toimus ilma veateadeteta.

Andmebaasiühenduse kohandamine:

Kui kasutaja PostgreSQL kasutajanimi või parool erineb seadistusest (postgres / salasona), tuleb avada järgmised .php failid ja muuta andmebaasi ühenduse stringi:

```
pg_connect("host=localhost dbname=arhnurk user=postgres
password=salasona");
```

Muuda `user=` ja `password=` vastavalt oma PostgreSQL kasutaja andmetele.

Failid, kus ühendus esineb:

- get_project.php
- load_gallery.php
- slider.php
- admin_dashboard.php
- admin_insert.php
- admin_delete.php
- admin_update.php
- admin_add.php
- admin_manage.php
- admin_slider.php
- admin_slider_insert.php
- admin_slider_update.php

- admin_slider_delete.php
- admin_edit.php

Veebiserveri käivitamine:

1. Käivita veebiserver (nt XAMPP)
2. Käivita veebiserveri kasutajaliideses Apache server.
3. Ava veebibrauser ja sisesta: http://localhost/Projekt/index.html

Haldusliidese kasutamine:

1. Ava URL: http://localhost/Projekt/admin/login.php
2. Logi sisse eeldefineeritud kasutajaga admin/sala.
3. Haldusliideses saab projekte lisada, muuta ja kustutada.

Litsents

Mina, Henri Tein,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Veebilehe loomine firmale Arhitektuurinurk OÜ“, mille juhendaja on Vambola Leping, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada Tartu Ülikooli digitaalarhiivi kuni autoriõiguse kehtivuse lõppemiseni;

2. annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi kaudu Creative Commonsi litsentsiga CC BY NC ND 4.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni;

3. olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile;

4. kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Henri Tein

12.05.2025