Pankkiautomaatti

TEKNINEN MÄÄRITTELY

Versio	1

Ryhmä nro	02
Valtteri Tenhunen	
Arttu Jämsä	
Juha Jermalainen	
Laura Similä	

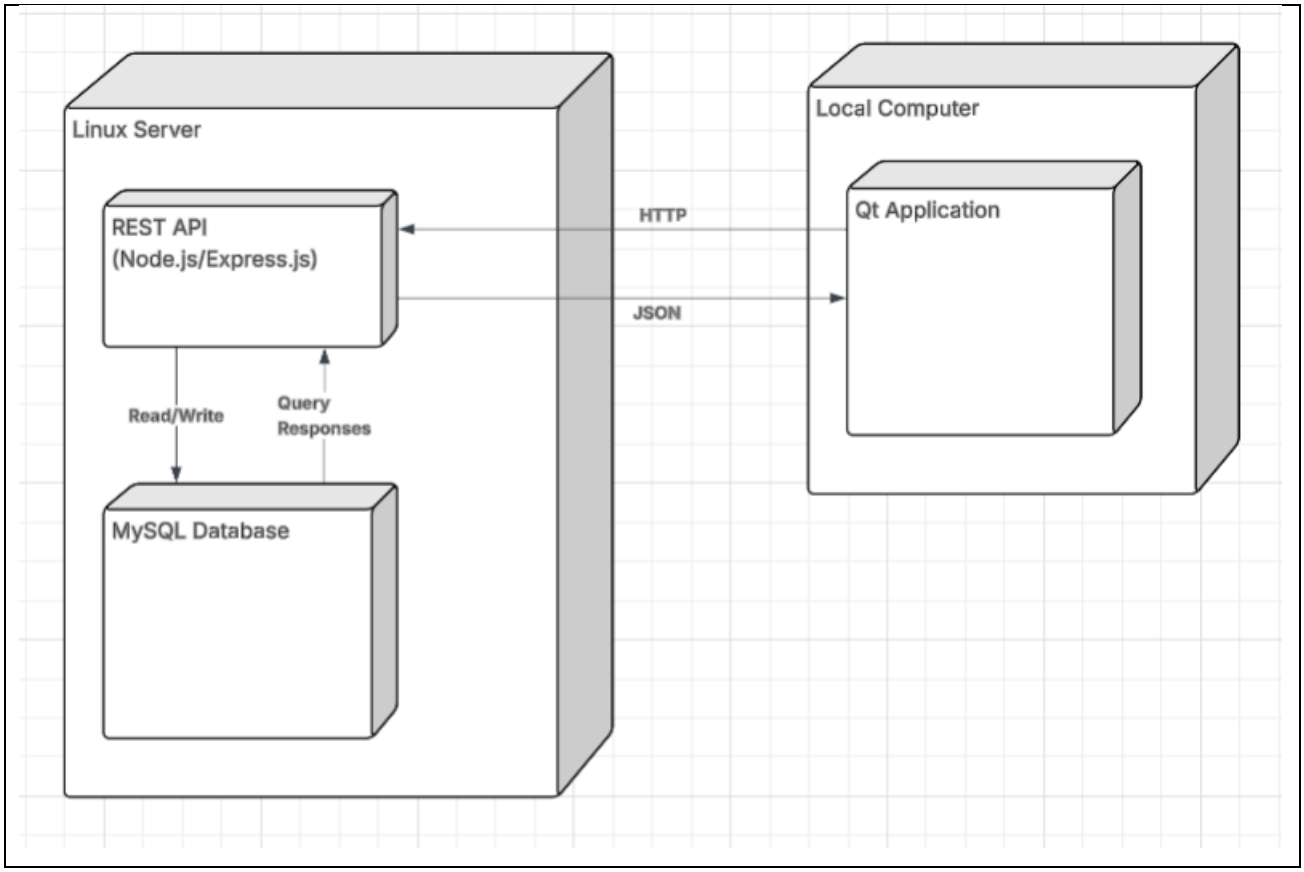
TEKNINEN MÄÄRITTELY

Dokumentti sisältää sekä toiminnallisen että teknisen määrittelyn toteutettavasta pankkiautomaattiohjelmistosta. Tarkoituksena on esittää:
1) Ohjelmistoon toteutettavat pankkiautomaatin käyttötapaukset, niistä johdetut toiminnallisuudet ja vaaditut ominaisuudet sekä näistä johdettu pankkiautomaatin tietomalli
2) Järjestelmäarkkitehtuuri, järjestelmän komponentit ja toteutuksen ratkaisut komponenteittain. Dokumentin tulee sisältää kuvaukset seuraavista: pankkiautomaattisovellus, mahdolliset DLL:t, REST API, tietokanta ja mahdolliset muut komponentit.
3) Pankkiautomaatin käyttöliittymän lopullinen toteutus kuvakaappauksilla ja tilakaaviolla.

Dokumentissa käytetään tarkoituksenmukaisia UML-mallinnuskielen kaavioita. Tietokanta ja sen tietomalli kuvataan ER-kaavion avulla.

JÄRJESTELMÄARKKITEHTUURI

Kuvassa alla esitetään projektissa kehitettävän pankkiautomaatin ohjelmiston järjestelmäarkkitehtuuri UML-mallinnuskielen käyttöönottokaavion avulla.



Yleiskuvaus

Pankkiautomaatti tarvitsee kohdejärjestelmän tietokoneessa toimiakseen tuoreen Windows-käyttöjärjestelmäversion ja tietoturvaohjelman (esim. Windows Defender).

Projektissa toteutettava pankkiautomaattiohjelmisto koostuu kolmesta järjestelmätason komponentista:

1. Käyttöliittymän toteuttavasta Windows-pohjaisesta pankkiautomaatti-sovelluksesta (ns. EXE-komponentti), joka hyödyntää osassa toiminnallisuuksia ajonaikaisesti ladattavia kirjastokomponentteja (DLL-komponentit). Nämä ohjelmistokomponentit toteutaaan Qt-

ohjelmointikehykseen perustuen, hyödyntäen Qt-luokkakirjaston valmiita käyttöliittymäkomponentteja sekä tapahtumapohjaista sovelluskehitystä.
2. Pankkiautomaattisovellus kommunikoi tietokannan kanssa HTTP-protokollaa käyttäen REST-pohjaisen verkkorajapinnan (REST API-komponentti) kautta. REST API toteutetaan node.js-ajoympäristön päälle hyödyntäen JavaScript-kielen ohjelmistokehystä express.js.
3. Tietokannan vaatima palvelinohjelma perustuu MySQL-tietokantaratkaisuihin.

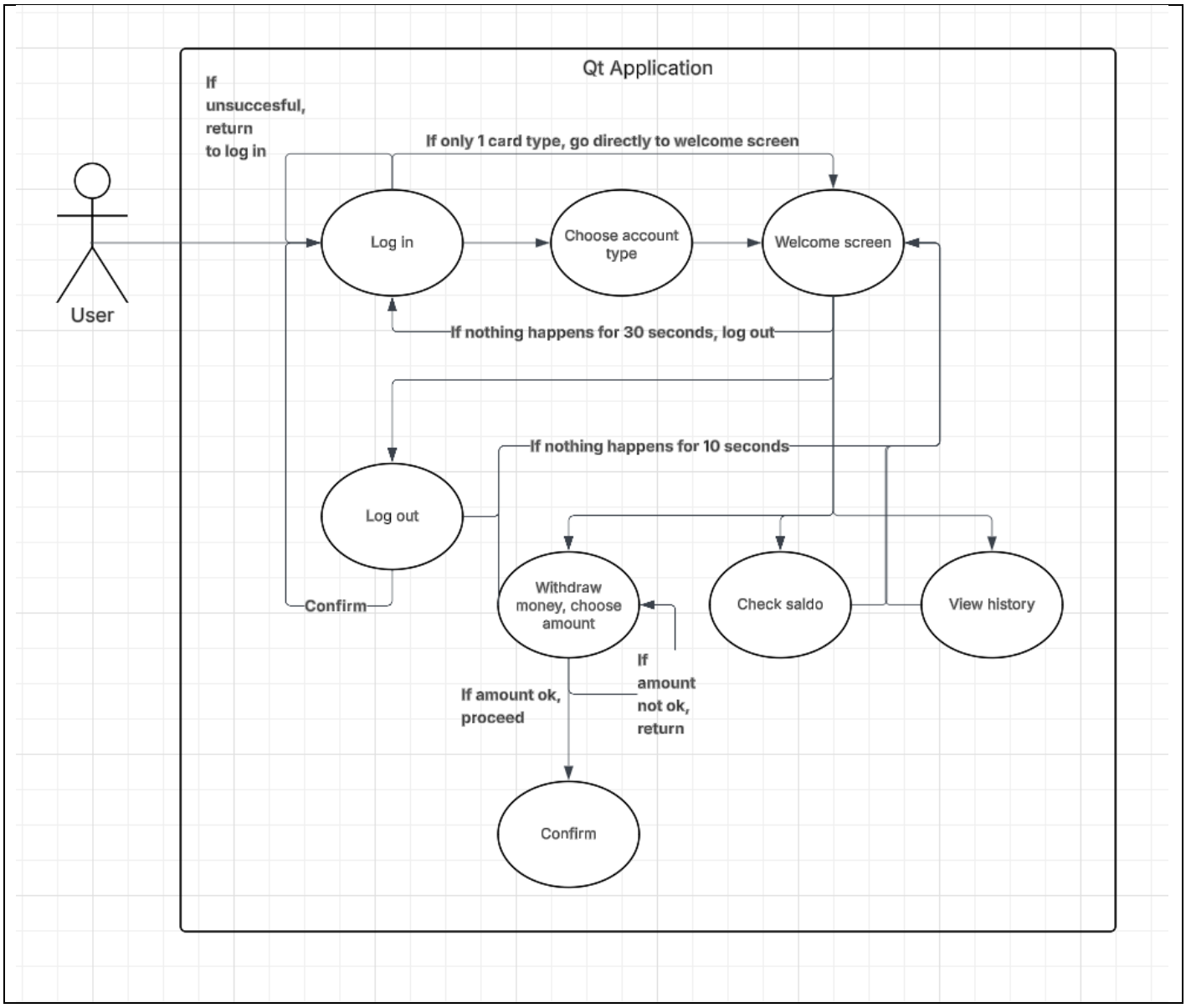
Pankkiautomaatin käyttäjän täytyy omistaa pankkikortti, joka on liitetty pankin tiliin. Kun kortin haltijalla on tiedossa kortin liitetty tunnusluku, hän voi käyttää pankkiautomaattia.

KÄYTTÖTAPAUKSET

Pankkiautomaattiohjelmiston keskeiset toiminnot on lueteltu alla. Näistä toiminnoista johdetaan UML-mallinnuskielen käyttötapauskaavio(t) ja jokaisesta käyttötapauksesta käyttötapauskortti.

TOIMINNON NIMI	TOIMINNON KUVAUS
Kirjaudu sisään	Kirjautua pankkiautomaatin käyttäjäksi kortin numeron ja tunnusluvun avulla.
Tilin valinta	Valitaan tili, jos kortilla sekä credit että debit
Näytä saldo	Näyttää tilin omistajan tiedot ja tilin saldon.
Selaa tilitapahtumia	Näyttää näytöllä tilitapahtumia käyttäjän selausvalintojen mukaisesti. 10 tapahtumaa kerrallaan
Nosta rahaa	Pankkiautomaatti luovuttaa käyttäjälle hänen nostaman summan rahaa, ja vähentää nostetun rahamäärän käyttäjän tililtä.
Kirjaudu ulos	Kirjata pankkiautomaatin käyttäjä ulos automaatista.
Kuvan lataus ja näyttäminen	Kuvan lataaminen backendiin ja näyttäminen Qt-sovelluksessa. Lisäominaisuus

Järjestelmän käyttötapauskaavio on allaolevan kuvan mukainen.


Qt Application
If unsuccesful, return to log in
If only 1 card type, go directly to welcome screen
User
Log in
Choose account type
Welcome screen
If nothing happens for 30 seconds, log out
If nothing happens for 10 seconds
Log out
Confirm
Withdraw money, choose amount
Check saldo
View history
If amount ok, proceed
If amount not ok, return
Confirm

Käyttötapauskortit

Käyttötapauskorteilla määritellään tarkemmin mitä jokaisessa toiminnossa tulee tapahtua. Käyttötapauskorteista voidaan näin ollen johtaa pankkiautomaattijärjestelmän toiminnalliset vaatimukset.

Nimi	**Kirjaudu sisään**
Suorittajat	Pankkiautomaattisovellus
Tavoite	Kirjautua pankkiautomaatin käyttäjäksi
Esiehdot	Automaatin tietokone toimii, yhteys tietokantaan on kunnossa ja tietokanta on toiminnassa
Kuvaus	1. Ohjelman aloitus -käyttöliittymä on esillä, jos automaattia ei käytetä. Siinä pyydetään käyttäjältä kortin numero ja PIN 2. Ohjelmassa avautuu tunnuksen syöttö -käyttöliittymä, jossa pyydetään käyttäjää syöttämään 4 numeroinen tunnusluku 2.1 Jos käyttäjä ei syötä mitään numeroita 10 sekunnin sisällä palataan takaisin aloitus-käyttöliittymään. 3. Kun tunnusluku on syötetty, niin kortin ID numero ja tunnusluku tarkistetaan tietokannasta. 3.1 Jos tunnusluku ei vastaa kortin ID-numeroa tietokannassa, niin siitä ilmoitetaan käyttäjälle. 3.2 Jos käyttäjä syöttää tunnusluvun kolme kertaa väärin, kortti lukitaan ja sitä ei voi enää käyttää. Tästä ilmoitetaan käyttäjälle, jonka jälkeen palataan ohjelman aloitus -käyttöliittymään. 4. Jos kortin ID numeroa vastaava tunnusluku syötettiin oikein, niin ohjelman pääkäyttöliittymä avautuu. 5. Pääkäyttöliittymässä näytetään korttiin liitetyn asiakkaan nimi, sekä voidaan valita vaihtoehdot: nosta rahaa, näytä saldo, selaa tilitapahtumia tai kirjaudu ulos. 6. Jos käyttäjä ei tee pääkäyttöliittymässä mitään 30 sekuntiin käyttöliittymä sulkeutuu, yhteydet tietokantaan suljetaan ja palataan aloituskäyttöliittymään.
Loppuehdot	Käyttäjä on kirjautunut järjestelmän käyttäjäksi.

Nimi	**Näytä saldo**
Suorittajat	Pankkiautomaattisovellus

Tavoite	Näyttää tilin omistajan tiedot ja tilin saldon
Esiehdot	Kirjauduttu pankkiautomaatin käyttäjäksi, yhteys tietokantaan on kunnossa ja tietokanta on toiminnassa
Kuvaus	1. Pääkäyttöliittymässä painetaan Näytä saldo –painiketta 2. Tietokannasta haetaan tiedot ja käyttöliittymässä näytetään tilin omistajan tiedot ja tilin saldo. 3. Käyttöliittymän Sulje-painiketta painamalla voidaan palata takaisin pääkäyttöliittymään. 4. Näytä saldo käyttöliittymä sulkeutuu ja palataan pääkäyttöliittymään, jos mitään painiketta ei paineta 10 sekuntiin. 5. Jos käyttäjä ei tee pääkäyttöliittymässä mitään 30 sekuntiin käyttöliittymä sulkeutuu, yhteydet tietokantaan suljetaan ja palataan aloituskäyttöliittymään.
Loppuehdot	Tilin omistajan tiedot ja saldo on näytetty oikein.

Nimi	**Selaa tilitapahtumia**
Suorittajat	Pankkiautomaattisovellus
Tavoite	Näytetään näytöllä 10 viimeisintä tilitapahtumaa käyttäjän selausvalintojen mukaisesti.
Esiehdot	Kirjauduttu pankkiautomaatin käyttäjäksi, yhteys tietokantaan toimii ja tietokanta on toiminnassa.
Kuvaus	1. Pääkäyttöliittymässä painetaan Selaa tilitapahtumia –painiketta 2. Tietokannasta haetaan tiedot ja käyttöliittymässä näytetään tilin omistajan tiedot ja 10 viimeistä tilitapahtumaa. 3. Tilitapahtumia voi selata painikkeilla eteen- ja taaksepäin siten, että aina siirrytään 10 tapahtumaan sen mukaan mitä painiketta painettiin. 4. Käyttöliittymän Sulje-painiketta painamalla voidaan palata takaisin pääkäyttöliittymään. 5. Jos mitään painiketta ei paineta 10 sekuntiin Selaa tilitapahtumia käyttöliittymä sulkeutuu ja palataan pääkäyttöliittymään. 6. Jos käyttäjä ei tee pääkäyttöliittymässä mitään 30 sekuntiin käyttöliittymä sulkeutuu, yhteydet tietokantaan suljetaan ja palataan aloituskäyttöliittymään.
Loppuehdot	Tilin omistajan tiedot ja tilitapahtumat on näytetty oikein, ja tilitapahtumia voidaan selata.

Nimi	**Nosta rahaa**
Suorittajat	Pankkiautomaattisovellus
Tavoite	Pankkiautomaatti luovuttaa käyttäjälle hänen nostaman summan rahaa, ja vähentää nostetun rahamäärän käyttäjän tililtä.
Esiehdot	Kirjauduttu pankkiautomaatin käyttäjäksi, yhteys tietokantaan on kunnossa ja tietokanta on toiminnassa.
Kuvaus	1. Pääkäyttöliittymässä painetaan Nosta rahaa –painiketta. 2. Tietokannasta haetaan tiedot ja käyttöliittymässä näytetään tilin omistajan tiedot, tilin saldo ja nostettavien rahamäärien painikkeet, esimerkiksi 20e, 40e, 60e, 100e ja muu summa 3. Käyttäjä painaa painiketta, jolla nostetaan painikkeen mukainen rahamäärä automaatista ja käyttäjän tilitä veloitetaan noston mukainen rahamäärä. 3.1 Debit tilillä ei ollut tarpeeksi rahaa, joten käyttäjälle ilmoitetaan ohjelman käyttöliittymässä tästä. 3.2 Credit tilillä ei ollut tarpeeksi luottorajaa jäljellä, joten käyttäjälle ilmoitetaan ohjelman käyttöliittymässä tästä. 4. Käyttöliittymän Sulje-painiketta painamalla voidaan palata takaisin pääkäyttöliittymään.
Loppuehdot	Käyttäjä on saanut nostetuksi haluamansa summan rahaa, ja rahamäärä on veloitettu käyttäjän tilitä.

Nimi	**Kirjaudu ulos**
Suorittajat	Pankkiautomaattisovellus
Tavoite	Lopettaa pankkiautomaatin käyttäminen ja kirjautua ulos järjestelmästä.
Esiehdot	Kirjauduttu pankkiautomaatin käyttäjäksi, yhteys tietokantaan on kunnossa ja tietokanta on toiminnassa.
Kuvaus	1. Pääkäyttöliittymässä painetaan Kirjaudu ulos –painiketta 2. Tietokantayhteys suljetaan ja käyttäjä kirjataan ulos pankkiautomaatista. 3. Palataan ohjelman aloituskäyttöliittymään.
Loppuehdot	Pankkiautomaatin yhteys tietokantaan on suljettu, käyttäjä on kirjattu ulos automaatista.

Nimi	**Kuvan lataus ja näyttäminen**
Suorittajat	Pankkiautomaattisovellus
Tavoite	Käyttäjälle näytetään tiliin liitetty kuva ja käyttäjä voi sitä vaihtaa. Joko oletuskuviin tai lisäämällä oman
Esiehdot	Kirjauduttu pankkiautomaatin käyttäjäksi, yhteys tietokantaan on kunnossa ja tietokanta on toiminnassa.
Kuvaus	1. Tilinvalintaruudussa näkyy käytössä oleva kuva sekä painike kuvan vaihtamiseen. 2. Painiketta painamalla avautuu ikkuna, jossa: 2.1 Voi valita oletuskuvista yhden 2.2 Ladata oman kuvan 2.3 Lukita valinnan 2.4 Palata tilinvalintaruutuun
Loppuehdot	Valittu kuva näytetään oikein ja päivitetään tietokantaan.

Nimi	**Backend CI/CD -putki**
Suorittajat	GitHub Actions, taustapalvelin (SSH + PM2)
Tavoite	Testata backend-koodi, tarkistaa koodin laatu ja riippuvuuksien turvallisuus sekä onnistuneen ajon jälkeen päivittää palvelinympäristö uusimmalla versiolla.
Esiehdot	- Koodi sijaitsee *backend/*-hakemistossa - GitHub Secrets sisältää kaikki palvelimen ja tietokannan salaisuudet - Palvelimella pyörii PM2 ja projektihakemisto on olemassa
Kuvaus	1. Workflow käynnistyy *push* tai *pull request → main* ja koskee backend-hakemistoa. 2. Node 20 asennetaan ja riippuvuudet haetaan komennolla `npm ci`. 3. Suoritetaan riippuvuusturvatarkistus `npm audit`. 4. Ajetaan testit (`npm test`). 5. Tarkistetaan löytyykö lint-komentoa ja ajetaan se jos mahdollista. 6. Jos kaikki onnistuu ja kyseessä on push → main, siirrytään deploy-vaiheeseen. 7. Backend päivitetään palvelimelle SSH:n kautta: `git pull`, `npm ci`, `.env` luodaan GitHub-secreteistä ja PM2 prosessi uudelleenkäynnistetään.

Loppuehdot	- Testit, audit ja lint ovat onnistuneet - Palvelimen backend-koodi päivitetty - PM2-prosessi käynnistetty uusilla ympäristömuuttujilla
Poikkeukset	- **Lint-komentoa ei löydy**: workflow ohittaa lint-vaiheen ja jatkaa normaalisti. - **npm audit löytää vain low-risk ongelmia**: ei estä putkea (audit-level: moderate). - **Testit epäonnistuvat**: deploy-vaihetta ei ajeta. - **SSH-yhteys epäonnistuu**: deploy epäonnistuu, palvelin ei päivity. - **GitHub Secrets puuttuu**: deploy-vaihe kaatuu environment-muuttujien luonnissa.

Nimi	Frontend CI/CD -putki
Suorittajat	GitHub Actions, Qt-työkalut, SSH-siirto etäpalvelimelle
Tavoite	Rakentaa Qt-pohjainen frontend-sovellus CI-ympäristössä, ajaa testit ja onnistuneen ajon jälkeen toimittaa valmis binääri palvelimelle.
Esiehdot	- Koodi sijaitsee *bank-automat/*-hakemistossa - GitHub Secrets sisältää palvelimen tiedot ja yksityisen SSH-avaimen - Qt 6.6.3 voidaan asentaa Linux CI-ympäristöön
Kuvaus	1. Workflow käynnistyy *push* ja *pull request* → main, jos muutos koskee frontend-hakemistoa tai CI-tiedostoa. 2. Qt 6.6.3 ja tarvittavat Linux-kirjastoriippuvuudet asennetaan aqtinstallilla ja apt-paketeilla. 3. CMake + Ninja -rakennusympäristö valmistellaan. 4. Sovellus konfiguroidaan ja käännetään Release-tilassa. 5. Testit ajetaan headless-tilassa Xvfb:n kautta. 6. Jos kyseessä push → main, valmis binääri siirretään palvelimelle SCP-actionilla.
Loppuehdot	- Frontend-sovellus on koottu ja testattu - Binääri siirretty palvelimelle oikeaan hakemistoon
Poikkeukset	- **Qt-asennus epäonnistuu**: workflow keskeytyy ennen build-vaihetta. - **Jokin Linux-kirjastoriippuvuus puuttuu**: CMake-konfiguraatio ei käynnisty → putki keskeytyy. - **Testit epäonnistuvat**: deploy-vaihetta ei suoriteta. - **SCP-siirto epäonnistuu** (virheellinen SSH-avain / host): binääri ei päivity palvelimelle. - **BACKEND_IP-env puuttuu**: build onnistuu, mutta sovelluksen runtime-ympäristö voi olla virheellinen—CÏ/CD ei kuitenkaan keskeydy.

Nimi	Swagger / OpenAPI -dokumentaation käyttö
Suorittajat	Kehittäjä, testaaja, Pankkiautomaatti-järjestelmän REST API
Tavoite	Tarjota selkeä ja interaktiivinen dokumentaatio Pankkiautomaatti-järjestelmän API-rajapinnoille sekä mahdollistaa autentikointi ja testikutsujen tekeminen suoraan Swagger-UI:n kautta.
Esiehdot	- Swagger-UI on käynnissä projektissa - Palvelin palauttaa OpenAPI 3.0.3 mukaisen rajapintakuvauksen - Käyttäjä tuntee testitunnukset tai omistaa voimassa olevan kortin tunnisteen ja PIN-koodin
Kuvaus	1. Käyttäjä avaa Swagger-UI-dokumentaation (OpenAPI 3.0.3). 2. Käyttäjä kirjautuu sisään kutsumalla `/auth/login` ja syöttämällä kortin tunnuksen ja PIN-koodin. 3. Vastauksen `token`-kenttä kopioidaan. 4. Swaggerin "Authorize"-painikkeesta tallennetaan JWT-token Bearer-kenttään. 5. Autentikoinnin jälkeen kaikki suojatut endpointit (/atm, /users, /cards, /accounts, /log) ovat testattavissa. 6. Käyttäjä voi kutsua saldo-, loki-, nosto- ja hallinnointienpointeja roolinsa mukaisesti. 7. Admin-roolin endpointit kuten `/users`, `/cards` ja `/accounts` ovat käytettävissä ainoastaan admin-tunnuksilla.
Loppuehdot	- Käyttäjä on autentikoitu Swaggerissa ja pystyy testaamaan API-kutsuja rooliensa mukaisesti - API-viestintä toimii dokumentaation kautta - Virhekoodit ilmaisevat selkeästi ongelmat (401/403/404/400)
Poikkeukset	- Väärä kortti tai PIN → /auth/login palauttaa 401 (Invalid credentials). [openapi | Txt] - Tokenia ei annettu Authorize-kenttään → Kaikki suojatut endpointit palauttavat 401. - Käyttäjällä ei ole pääsyä toisen käyttäjän tietoihin → Palautuu 403 (Forbidden). Tämä koskee esim. /atm/{id} ja admin-rajapintoja. [openapi | Txt] - Tiliä, käyttäjää tai korttia ei löydy → Palautuu 404. [openapi | Txt] - Nostettava summa on virheellinen tai saldo ei riitä → /atm/{id}/withdraw

	palauttaa 400. [openapi | Txt] - Token vanhentuu tai se on väärässä muodossa → 401 Unauthorized kaikissa suojatuissa kutsuissa. - Admin-oikeuksia vaativiin kutsuihin käytetään user-tunnusta → 403 Forbidden (/users, /cards, /accounts, /log).

TIETOSISÄLTÖ

Tässä kuvataan pankkiautomaatiohjelman ja -järjestelmän käsittelemä tieto.

Käsiteanalyysi

Pankkiautomaattijärjestelmässä käsitellään tietoja seuraavien lähtöoletusten ollessa voimassa:

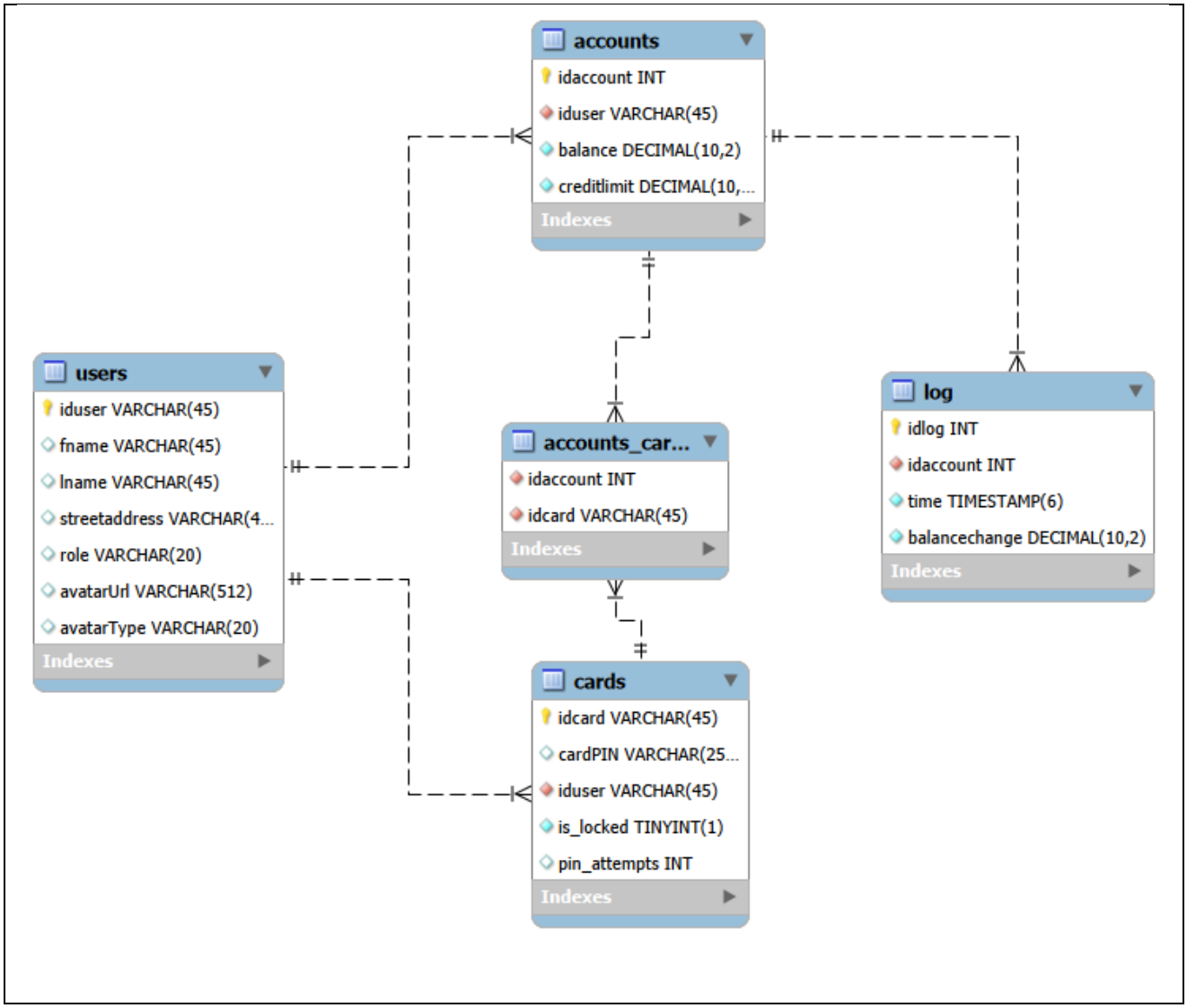
- Yksi kortti kuuluu yhdelle asiakkaalle
- Asiakkaalla voi olla monta tiliä tyyppiä debit tai credit
- Asiakkaalla voi olla monta korttia
- Tiliin voidaan liittää monta korttia
- Kortin tyyppejä on kolme: debit, credit, kaksoiskortti
- Yksi kortti voidaan liittää useaan tiliin (tämä tarkoittaa käytännössä, että kaksoiskortti voidaan liittää yhteen debit-tiliin ja yhteen credit-tiliin)

Pankkiautomaattijärjestelmässä käsitellään seuraavia tietoja:

Asiakas - Asiakkaan tunnus - Asiakkaan nimi - Asiakkaan lähiosoite - Kuvan url - Kuvan tyyppi - Rooli, user tai admin	**Tili** - Tilinumero - Tilin saldo - Tilin omistaja - Luottoraja
Kortti - Kortinnumero - Kortin PIN-koodi - Korin omistaja - Onko lukittu - Pin yritykset	**Tilitapahtumat** - Tilinumero - Päivämäärä ja kellonaika - Tapahtuma - Summa

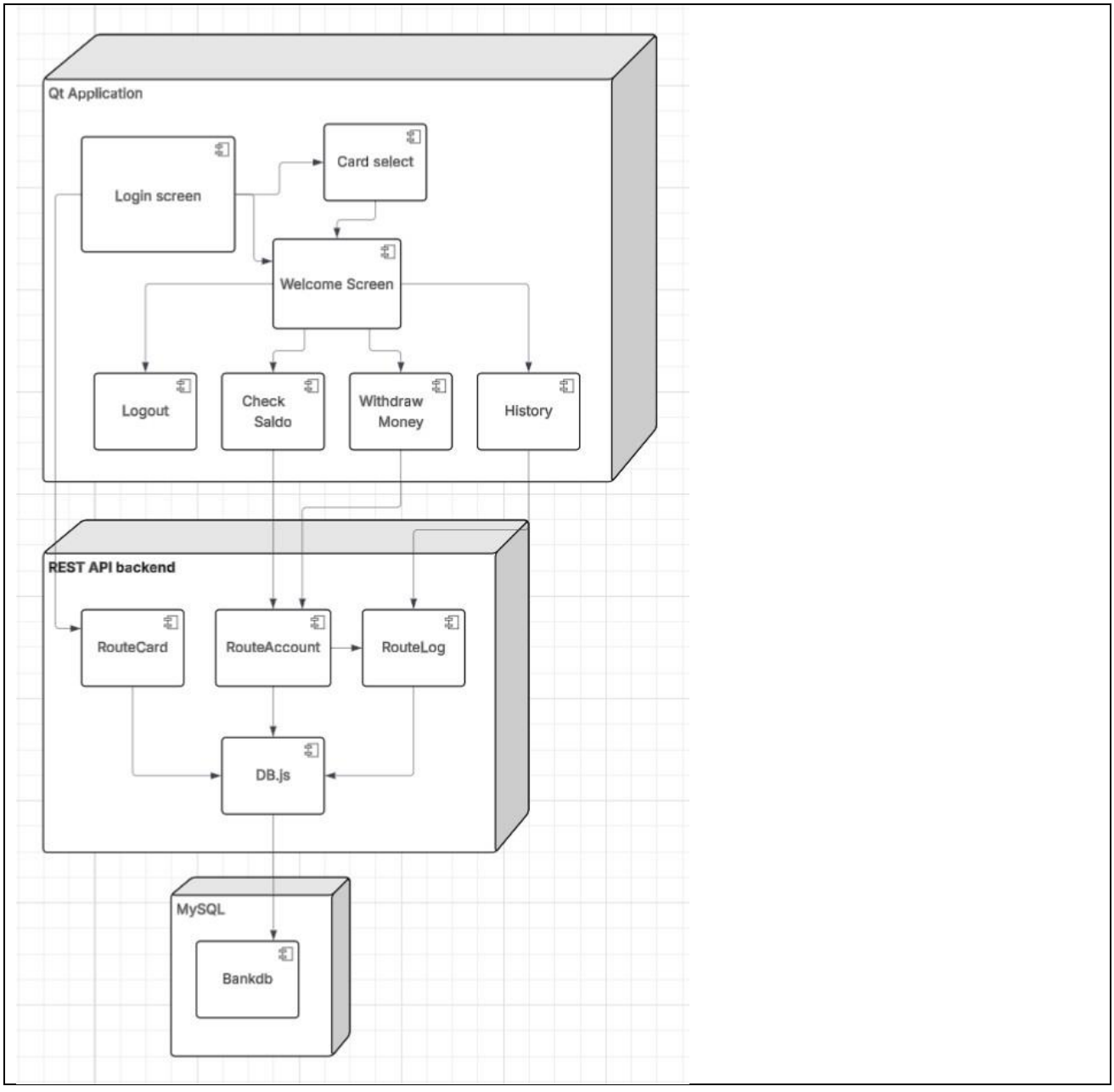
Käsitemalli

Käsiteanalyysin perusteella laadittu tietokannan rakennetta kuvaava ER-mallikaavio liitetään tähän.



JÄRJESTELMÄN KOMPONENTIT

Tässä esitetään ensin järjestelmäarkkitehtuuria tarkentava UML-komponenttikaavio. Kaaviossa kuvataan miten järjestelmäkomponentit toteutetaan ohjelmistokomponentten avulla, mitkä ovat niiden käyttämät rajapinnat ja muut yhteydet.

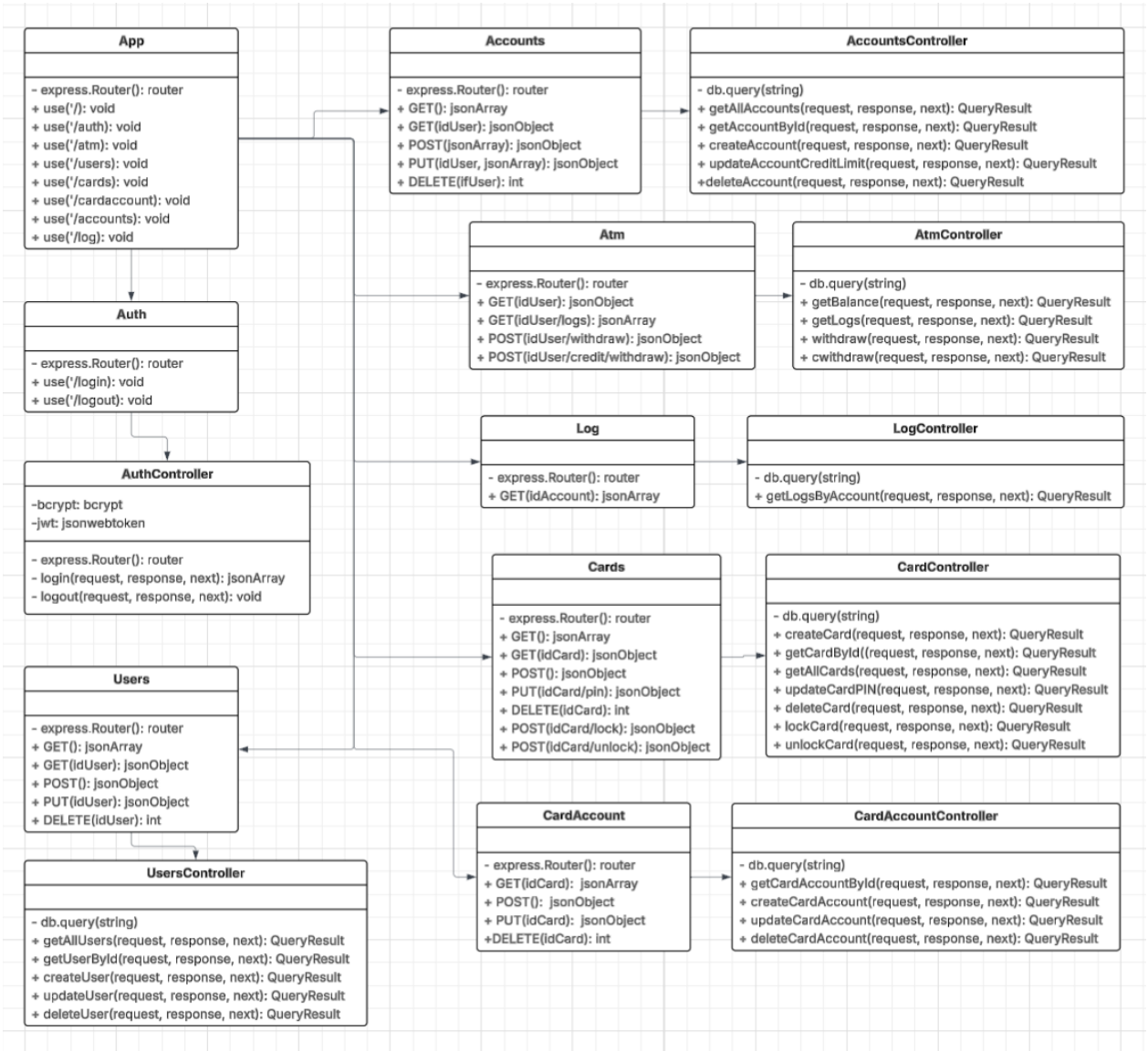


KOMPONENTTIEN KUVAUKSET

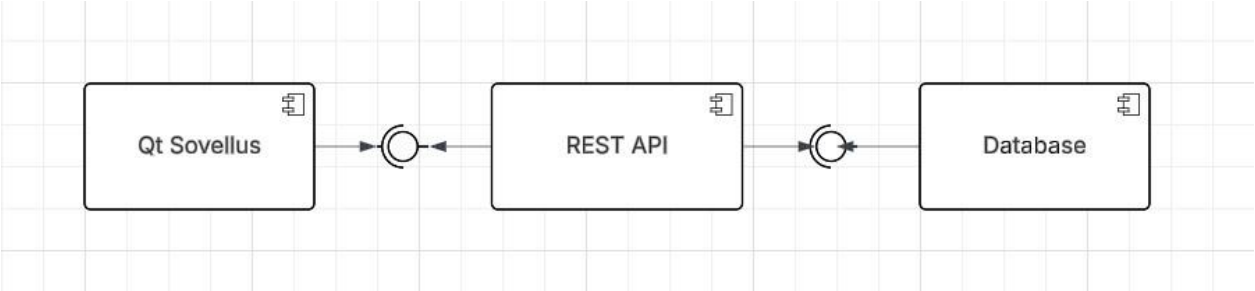
REST API

Tarkoitus ja toiminta	Kommunikointiin frontend Qt sovelluksen ja databasen välille.
Järjestelmäkomponentti	REST API

Luokkakaavio (UML)



Rajapinnat



API Contract v2 – Bank ATM REST API

1) General

- **Protocol:** HTTP
- **Data format:** JSON
- **Base URL (dev):** http://localhost:3000
- **Content-Type:** `application/json`
- **Authentication:** JWT
- **Swagger UI:** `GET /docs`

2) Authentication

2.1 Login

POST `/auth/login`

Request

```
{
  "idCard": "CARD123456",
  "pin": "1234"
}
```

Response (200 OK)

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
```

```
  "card": {
    "idCard": "CARD123456",
    "idUser": "TESTUSER1",
    "isLocked": false
  },
  "accounts": [
    {
      "idAccount": 14,
      "type": "debit",
      "balance": 500.00,
      "creditLimit": 0.00
    },
    {
      "idAccount": 15,
      "type": "credit",
      "balance": 0.00,
      "creditLimit": 2000.00
    }
  ],
  "requiresAccountSelection": true
}
```

2.2 Logout

POST `/auth/logout`

- Requires header: `Authorization: Bearer <token>`
- Token may be invalidated server-side (blacklist).

3) JWT Usage

All protected endpoints require:

`Authorization: Bearer <token>`

4) Roles and Permissions

The system supports the following roles:

- `user` (default): may only use ATM endpoints related to their own accounts
- `admin`: may access all administrative endpoints

Admin-only endpoints

- `/users/*`
- `/cards/*`
- `/accounts/*`
- `/cardaccount/*`
- `/log/*`

User endpoints

- `/atm/*`

(requires authentication and ownership of the account)

5) HTTP Status Codes

Situation	Status
Successful GET	200 OK
Resource created	201 Created
Successful DELETE	204 No Content
Invalid or missing data	400 Bad Request
Not authenticated	401 Unauthorized
Not authorized	403 Forbidden
Resource not found	404 Not Found
Conflict / duplicate resource	409 Conflict
Unexpected server error	500 Internal Server Error

6) REST API Endpoints

6.1 ATM Operations (User)

Requires JWT authentication and ownership of the account.

Get account balance

GET `/atm/:id`

Response (200 OK)

```
{
  "idAccount": 14,
  "idUser": "TESTUSER1",
  "balance": 500.00,
  "creditLimit": 0.00
}
```

Get account transaction history (ATM view)

GET `/atm/:id/logs`

Response (200 OK)

```
{
  "items": [
    {
      "idLog": 12,
      "time": "2026-01-16T09:30:00.123000",
      "balanceChange": -20.00
    }
  ]
}
```

Withdraw money (debit)

POST `/atm/:id/withdraw`

Request

```
{
  "amount": 40.00
}
```

Response (200 OK)

```
{
  "idAccount": 14,
  "balance": 460.00,
  "logged": true
}
```

Withdraw money (credit)

POST `/atm/:id/credit/withdraw`

Request

```
{
  "amount": 200.00
}
```

Response (200 OK)

```
{
  "idAccount": 15,
  "balance": -200.00,
  "logged": true
}
```

6.2 Users (Admin)

Requires JWT and `role: admin`.

Get user by ID

GET `/users/:idUser`

Response (200 OK)

```
{
  "idUser": "USER123",
  "firstName": "Matti",
  "lastName": "Meikäläinen",
```

```
  "streetAddress": "Example Street 1",
  "role": "user"
}
```

Create new user

POST `/users`

Request

```
{
  "idUser": "USER123",
  "firstName": "Matti",
  "lastName": "Meikäläinen",
  "streetAddress": "Example Street 1",
  "role": "user"
}
```

- `role` is optional, default is `"user"`
- Allowed values: `"user"`, `"admin"`

Response (201 Created)

```
{
  "idUser": "USER123",
  "firstName": "Matti",
  "lastName": "Meikäläinen",
  "streetAddress": "Example Street 1",
  "role": "user"
}
```

Update user

PUT `/users/:idUser`

Request

```
{
  "firstName": "Maija",
  "lastName": "Virtanen",
  "streetAddress": "New Street 5"
}
```

Response (200 OK)

```
{
  "idUser": "USER123",
  "firstName": "Maija",
  "lastName": "Virtanen",
  "streetAddress": "New Street 5"
}
```

Delete user

DELETE `/users/:idUser`

Response

- 204 No Content

6.3 Cards (Admin)

Requires JWT and `role: admin`.

Get all cards

GET `/cards`

Get card by ID

GET `/cards/:idCard`

Create card (PIN is hashed with bcrypt)

POST `/cards`

```
{
  "idCard": "CARD789012",
  "idUser": "USER123",
  "cardPIN": "1234"
```

```
}
```

Update card PIN

PUT `/cards/:idCard/pin`

Lock card

POST `/cards/:idCard/lock`

Unlock card

POST `/cards/:idCard/unlock`

Delete card

DELETE `/cards/:idCard`

6.4 Accounts (Admin)

Requires JWT and `role: admin`.

Get account

GET `/accounts/:id`

Create account

POST `/accounts`

Update credit limit

PUT `/accounts/:id`

Delete account

DELETE `/accounts/:id`

An account must not be linked to any cards when deleting.

6.5 Card–Account Links (Admin)

Requires JWT and `role: admin`.

Get linked accounts for a card

GET `/cardaccount/:idCard`

Link card to account

POST `/cardaccount`

Update card–account link

PUT `/cardaccount/:idCard`

Remove card–account link

DELETE `/cardaccount/:idCard`

6.6 Logs (Admin)

Requires JWT and `role: admin`.

Get account logs (newest first)

GET `/log/:idAccount`

7) Environment Variables

Create `backend/.env`:

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=your_password
DB_NAME=bank_db
DB_PORT=3306

JWT_SECRET=your-secret-key-here
PIN_PEPPER=your-pepper-here

PORT=3000
```

8) Database Initialization

```
cd backend/db
sudo mysql -u root bank_db < schema.sql
sudo mysql -u root bank_db < procedures.sql
sudo mysql -u root bank_db < seed.sql
```

9) Test Credentials

Regular user

- Card: `CARD123456`
- PIN: `1234`

Admin user

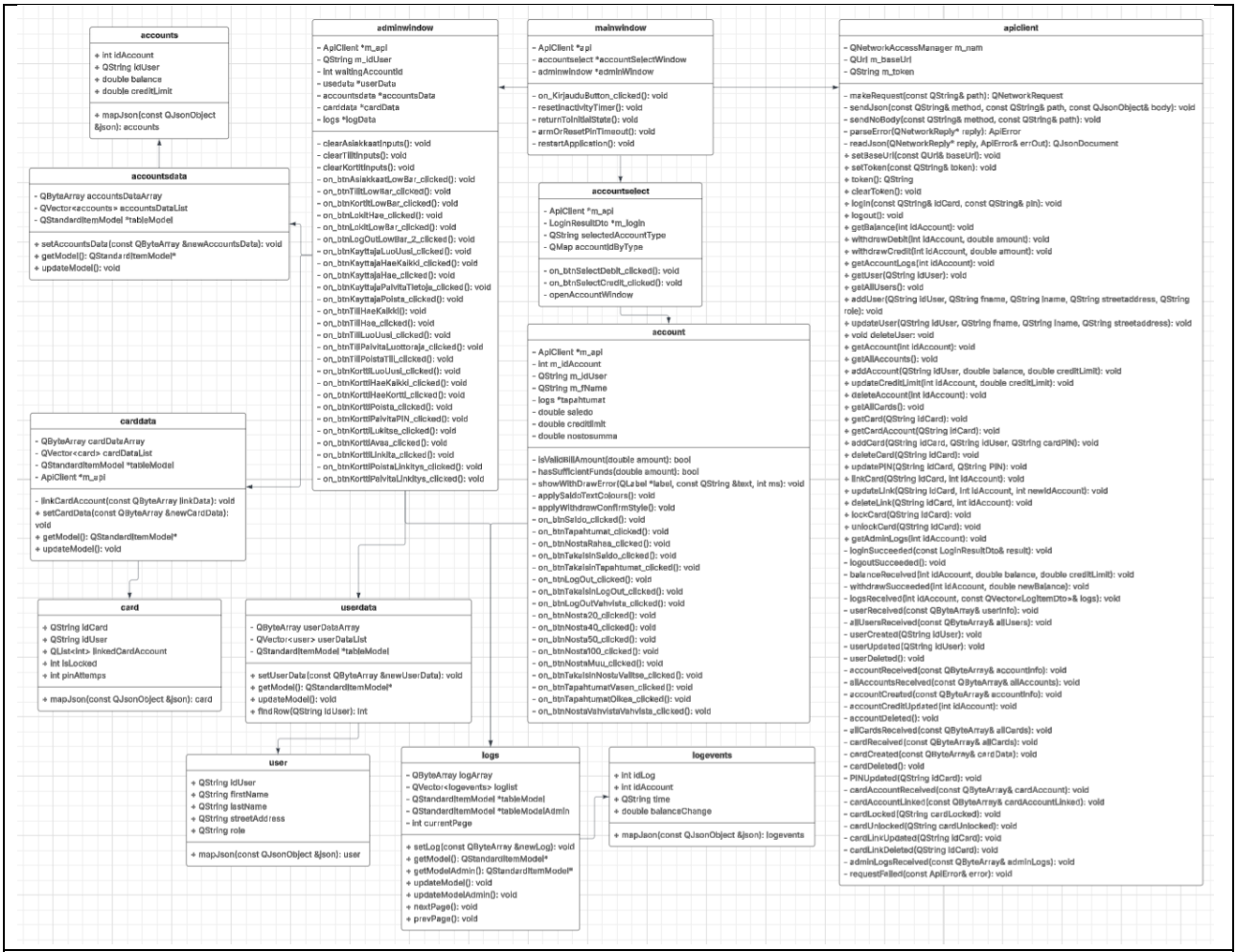
- Card: `ADMINCARD`
- PIN: `admin123`

Vastuuhenkilö(t)	Laura Similä: backend REST API (pari CRUDia ja controlleria, API-sopimus, README)

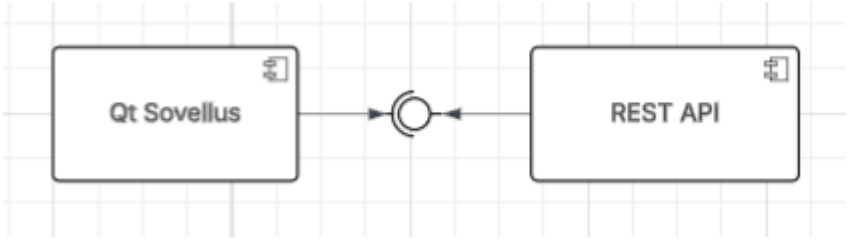
Qt sovellus

Tarkoitus ja toiminta	Käyttöliittymä koko kokonaisuudelle
Järjestelmäkomponentti	Sovellus

Luokkakaavio (UML)



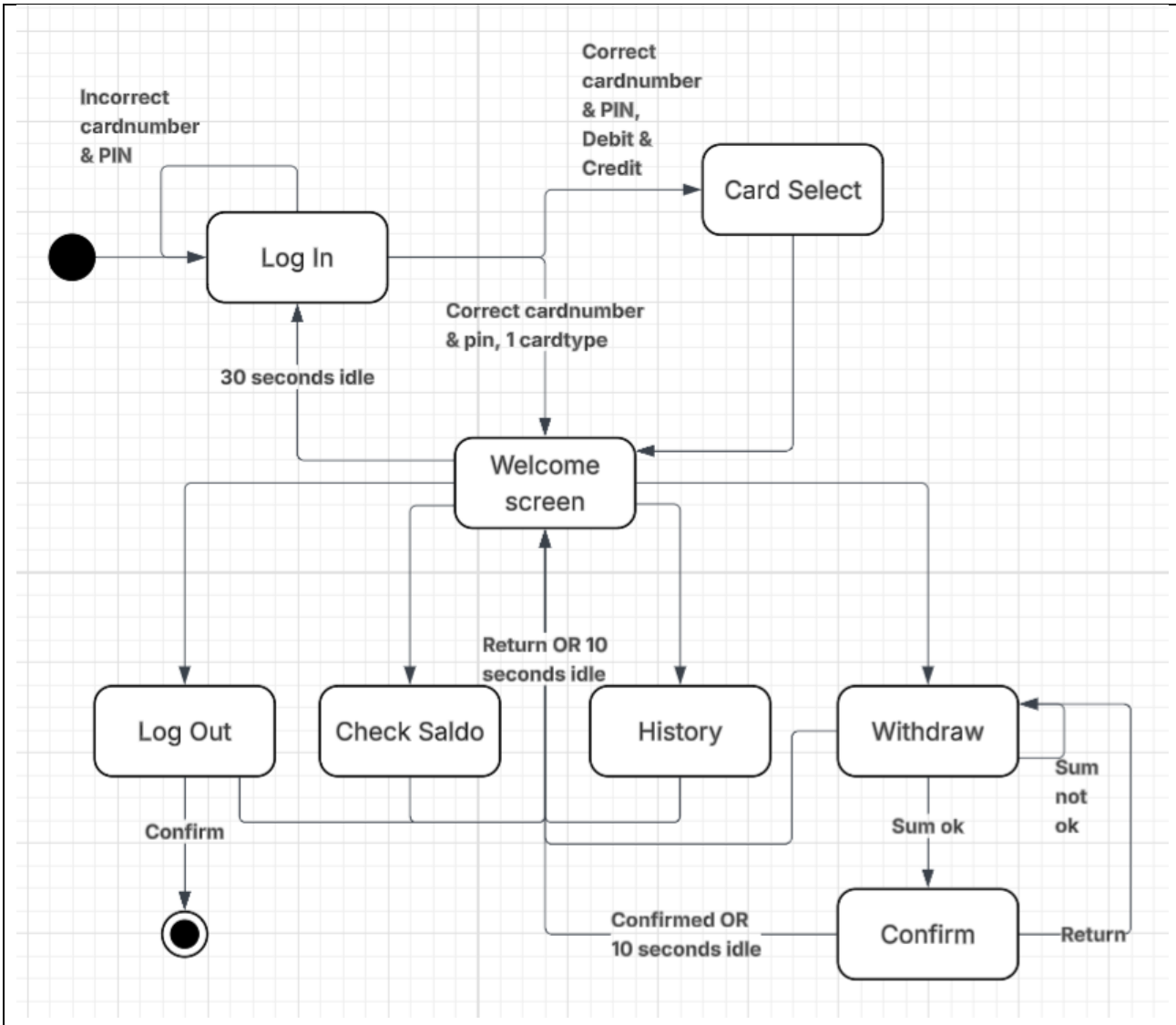
Rajapinnat



Funktio / Signaali	Tarkoitus
REST API	Käyttää REST APIa tietokantaan pääsyä varten

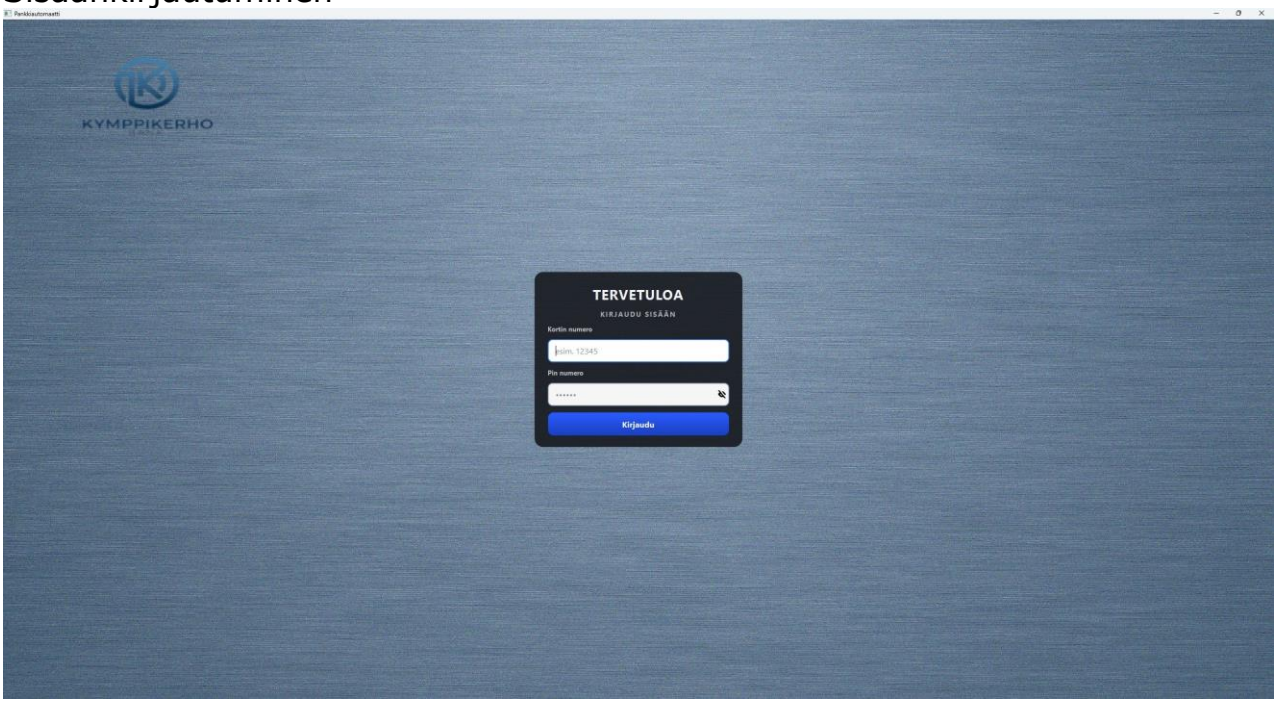
Vastuuhenkilö(t)	Juha Jermalainen: Pääikkuna Arttu Jämsä: UIn runko, adminpuoli ja dokumentointi Laura Similä: APIn toteutus (APIClient) ja ATM-puolen hiominen kuntoon, PIN ja global inactivity timeout Valtteri Tenhunen: Kuvien valinta ja lisäys

KÄYTTÖLIITTYMÄN KUVAUS



Ohjelman käyttöliittymät

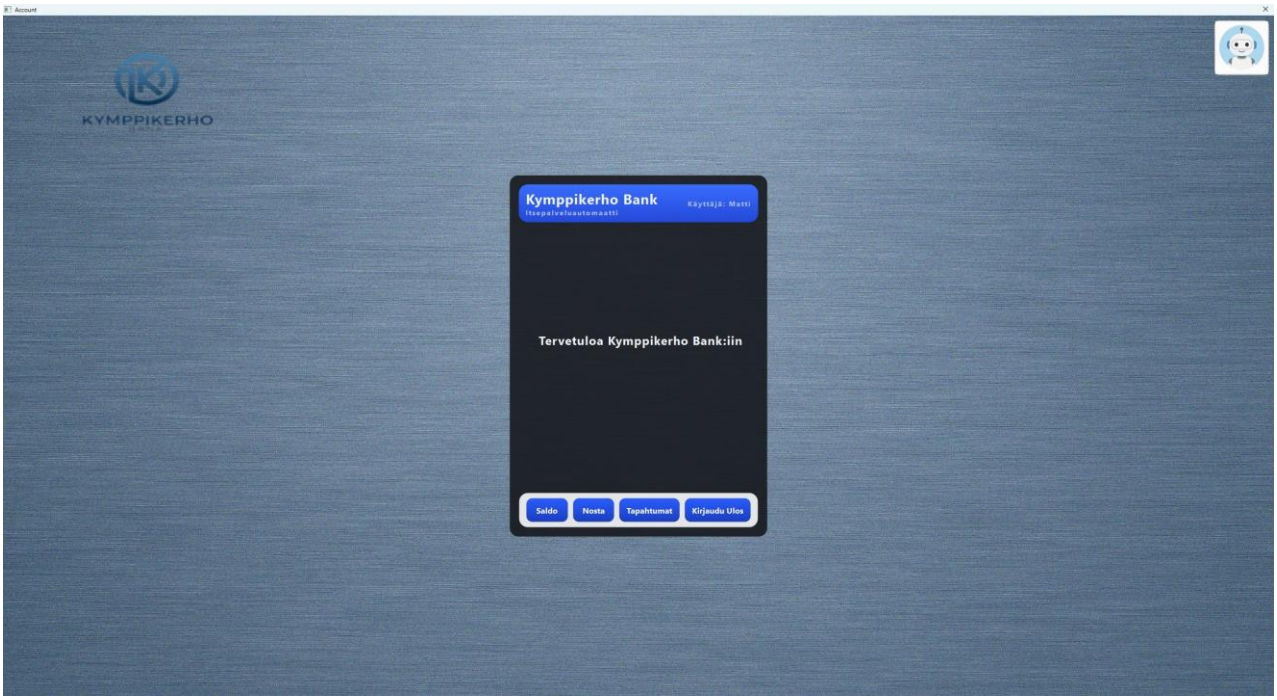
Sisäänkirjautuminen



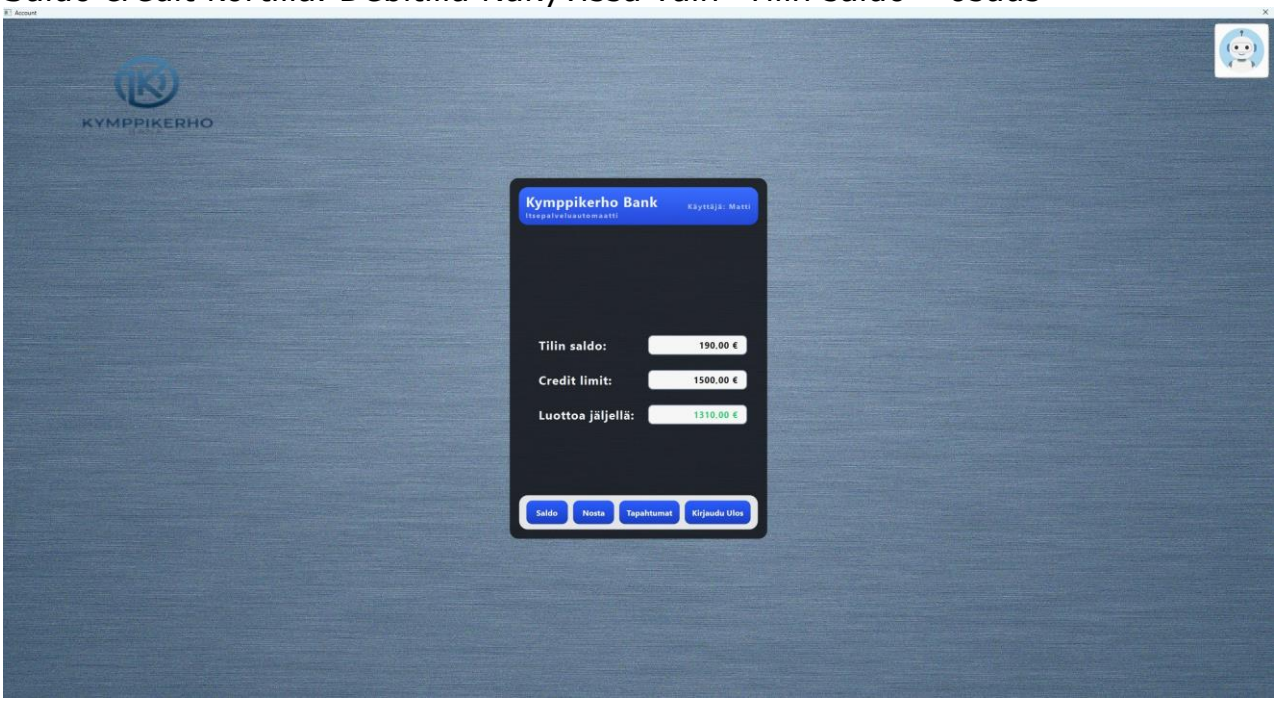
Tilin valinta

Kuvan vaihto

Tervetulo-ruutu

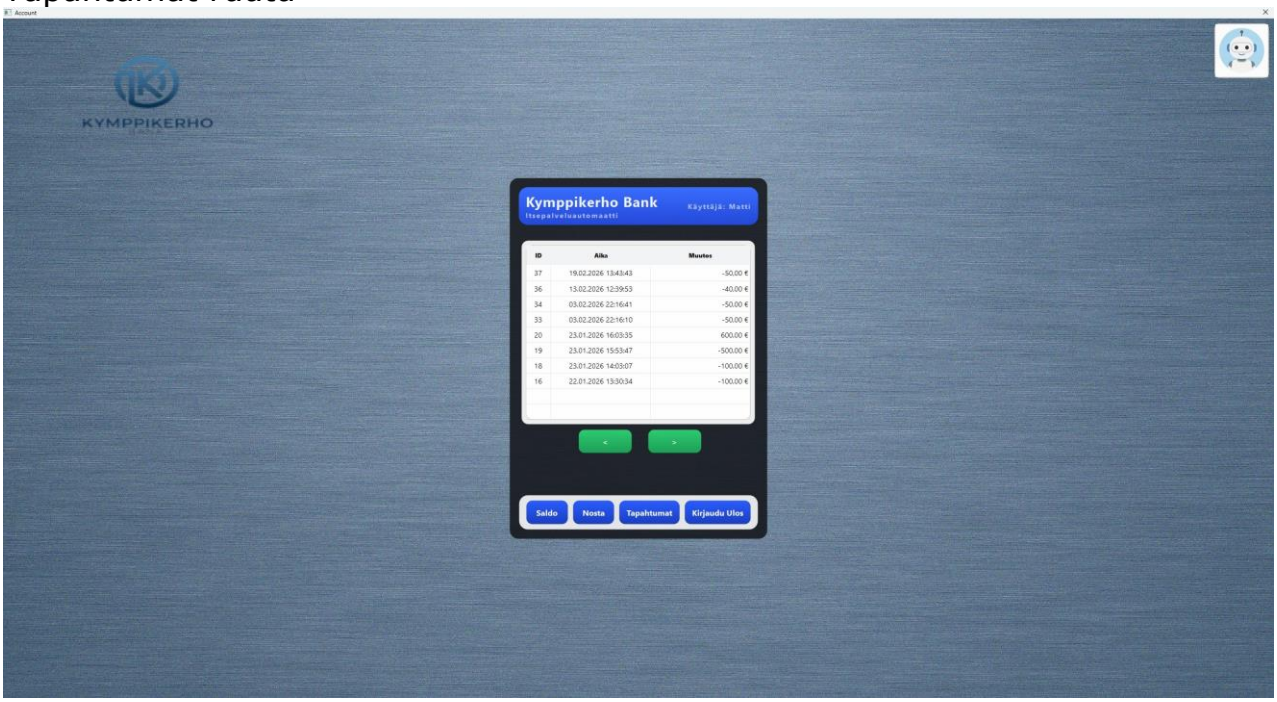


Saldo credit kortilla. Debitillä Näkyvissä vain “Tilin saldo” -osuus



Nosta rahaa ruutu

Tapahtumat ruutu



Uloskirjautuessa varmistus -ruutu