

TESTAUSSUUNNITELMA

Kipukalenteri

Jaana Pusa
pusa.jaana@gmail.com

Sisällysluettelo

1	Johdanto	2
1.1	Testauksen kohde ja tavoitteet.....	2
1.2	Testausympäristö.....	2
2	Testausstrategia	2
3	Testauksen vaiheet	3
3.1	Manuaalinen testaus	3
3.1.1	Tietokannan manuaalinen testaus SQL Serverissä	3
3.1.2	Kontrollerien manuaalinen testaus Postman-sovelluksella.....	11
3.1.3	Käyttöliittymän manuaalinen testaus.....	22
3.2	Automaattinen testaus	33
3.2.1	Yksikkötestaus Jest-kirjastolla.....	33
3.2.2	Integraatiotestaus Alba -kirjastolla.....	45
3.2.3	End-to-end testaus Cypress-kirjastolla	55
3.3	Hyväksymistestaus.....	60
3.3.1	Use case 1: Kipumerkinnän tallennus, muokkaus ja poisto	61
3.3.2	Use case 2: Kivun sijainnin lisäys.....	61
3.3.3	Use case 3: Erillisen muistiinpanon lisäys	62
3.3.4	Use case 4: Kipumerkintöjen suodatus.....	63
4	Vanhat tekstiosiot	64
5	Liitteet	65

1 Johdanto

Tässä projektissa tuotetaan Kipukalenteri -web-sovellus Careerian tieto- ja viestintäteknikaan ohjelmistotuotannon opiskelijan toimesta. Projektin on osa opiskelijan näyttötutkinnon suoritusta. Tämä dokumentti on sovelluksen testaussuunnitelma.

Kipukalenteri-sovelluksen testaussuunnitelmissa varmistetaan, että web-sovellus toimii dokumentoidulla tavalla eikä sovelluksesta puutu määrittelydokumentissa esitellyjä toteutettavaksi valittuja toimintoja. Testauksessa ei aina voida havaita kaikkia sovelluksen virheitä, mutta tavoitteena on tuottaa vähintään keskeisimmiltä osiltaan toimiva sovellus. Testaus suoritetaan loppukäyttäjän näkökulmasta.

Jotta testausprosessi on mahdollisimman hyvin suoritettu ja koska opiskelijan osaaminen automaattisesta testauksesta on rajallinen, testauksen pääpaino on manuaalisessa testauksessa. Manuaalisen testauksen jälkeen tehdään kuitenkin myös automaattiset testit.

1.1 Testauksen kohte ja tavoitteet

Tavoitteena on osoittaa, että sovellus toimii dokumentoitujen määrittelyjen mukaisesti sekä siinä on mukana määrittelydokumentissa esitellyt korkeamman prioriteetin toiminnot.

Testattavat toiminnot tulevat olemaan muun muassa:

- Onnistuuko sijainnin lisääminen, poistaminen ja muokkaaminen virheittä?
- Onnistuuko muistiinpanon lisääminen, poistaminen ja muokkaaminen virheittä?
- Onnistuuko kipumerkinnän lisääminen, poistaminen ja muokkaaminen virheittä?
- Miten sovellus/sovelluksen osat toimivat virheellistä tietoa syötettäessä?
- Toimivatko painikkeet oikein?
- Onko sovellukset tekstit asemoit oikein sekä mobiilissa että tietokoneella?
- Ovatko sovelluksen antamat ilmoitukset järkeviä?

1.2 Testausympäristö

Manuaalisen testauksen testausympäristönä toimivat SQL Server Management Studio, Postman-sovellus sekä lopuksi valmis Kipukalenteri -web-sovellus. Tietokannasta ja ASP.NET-Core API-projektista testataan manuaalisesti sovelluksen kannalta olennaiset toiminnot. Web-sovelluksen manuaalinen testaaminen tapahtuu simuloiden loppukäyttäjän toimintaa käyttötilanteessa. Testaus suoritetaan ensisijaisesti Windows-käyttöjärjestelmässä Chrome- ja Mozilla -selaimilla sekä android-puhelimella.

Automaattinen testaaminen suoritetaan react-projektissa Jest- ja Cypress-kirjastoilla ja ASP.NET Core API-projektissa Alba- ja xunit-kirjastoilla.

2 Testausstrategia

Testaaminen jakautuu kolmeen vaiheeseen ja etenee V-mallin mukaisesti aina yksittäisen osion tai toiminnon testaamisesta laajempien kokonaisuuksien tai useamman ohjelman osan yhtäaikaiseen testaamiseen. Testauksessa panostetaan erityisesti manuaaliseen testaamiseen ja testitapausten monipuolisuteen, jotta sovellus saadaan testattua mahdollisimman kattavasti.

Manuaalinen testaus aloitetaan tietokannan testauksella ja suoritetaan tietokannan rakentamisen jälkeen sekä mahdollisten tietokantamuutosten yhteydessä SQL Server Management Studiossa lisäämällä tietokantaan testidataa sekä suorittamalla tietokantahakuja

sekä testidatan muokkausta ja poistoja SQL-kielellä. Seuraavaksi testataan ASP.NET Core API:n ja tietokannan välinen toimivuus Postman-sovelluksen avulla sen jälkeen, kun back end – sovelluksen kontrollerit on saatu valmiiksi. Käyttöliittymän alustavat manuaaliset testit sovelluksen toimintojen ohjelmoinnin yhteydessä sekä lopuksi vielä sovelluksen valmistuttua.

Automaattinen testaaminen aloitetaan käyttöliittymän toimintojen valmistumisen jälkeen. Yksikkötestaus toteutetaan React-projektissa Jest-testauskirjastolla, integraatiotestaus ASP.NET Core API:ssa toteutetaan tämän jälkeen xunit ja Alba-kirjastoilla. Lopuksi sovelluksen valmistuttua toteutetaan end-to-end -testaus Cypress-kirjastolla React-projektissa.

Viimeisessä vaiheessa tehdään vielä hyväksymistestaus, jossa ohelman toimintoja verrataan määrittelykuvastossa esitettyihin vaatimuksiin.

3 Testauksen vaiheet

3.1 Manuaalinen testaus

Kipukalenteri-sovelluksen manuaalinen testaus aloitetaan tietokannan testaamisella SQL Server Management Studiossa. Seuraavaksi testataan ASP.NET Core API-kontrollerit Postman-sovelluksen avulla ja lopuksi testataan vielä toteutettu käyttöliittymä ja sen toiminnot valmiin web-sovelluksen kautta.

Manuaalisen testauksen lopputulos on käsitelty tämän dokumentin liitteenä olevassa testauspöytäkirjassa.

3.1.1 Tietokannan manuaalinen testaus SQL Serverissä

Tietokanta testataan manuaalisesti SQL-kielellä lisäämällä oikeellista, virheellistä ja puutteellista testidataa tietokantatauluihin sekä suorittamalla tietokantahakuja sekä yhdistämällä taulujen tietoja toisiinsa. Lisäksi muokataan ja poistetaan testidataa. Tietokannassa on testaushetkellä valmiina testidataa, jonka muokkautuminen ei haittaa. Testauksen kohteena ovat kaikki sovelluksen tietokantataulut eli Notes, PainLocation, Users ja PainLog.

Käyttöliittymässä mahdolistetaan vain yhden rivin lisääminen tai muokkaaminen kerrallaan, mutta testauksessa käydään läpi myös useamman rivin lisääminen ja muokkaaminen siltä varalta, että tietokantaan tätyy joskus lisätä tietoa manuaalisesti. Käyttöliittymässä myös rajoitetaan jonkin verran sitä, minkä tyypistä tietoa tietokenttiin on mahdollista syöttää, mutta tietokannan testauksessa tutkitaan myös tapauksia, joissa kyseisiin kenttiin syötetään väärän tyypistä tietoa.

3.1.1.1 Notes-taulun manuaalinen testaus

Notes-tauluun tallennetaan käyttäjän muistiinpanoja. Taulussa on ala olevat kentät. Pääavain NoteId tulee tietokannasta automaatisesti, sillä on identity-ominaisuus, joten tietoa ei syötetä/muokata/poisteta taulusta. UserId tulee tauluun automaatisesti käyttäjän käyttöliittymästä tekemän tiedon lisäyksen kautta, mutta tietokannan testauksen yhteydessä sen lisääminen testataan.

NoteId	Muistiinpanon id	int	primary key	Not null
NoteDate	Päivämäärä	datetime		Not null
NoteText	Teksti	nvarchar		Not null
UserId	Käyttäjätunnus	nvarchar	foreign key (Users-username)	Not null

3.1.1.1.1 Syötetään tauluun yksi rivi oikeellisia tietoja

Syötetään tauluun rivi, jossa on vähintään kaikki vaaditut tiedot oikeassa tietotyypissä.

```
Insert into Notes (NoteText, NoteDate, UserId) values ('SQL Server DB manual test 3.1.1.1A', '2022-06-03 10:57:00', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty rivi löytyy tietokantataulusta.

3.1.1.1.2 Syötetään tauluun useita rivejä oikeellisia tietoja

Syötetään tauluun kolme riviä, joissa on vähintään vaaditut tiedot oikeassa tietotyypissä.

```
Insert into Notes (NoteText, NoteDate, UserId) values ('SQL Server DB manual test 3.1.1.1B', '2022-06-03 11:57:00', 'testiad'), ('SQL Server DB manual test 3.1.1.1B', '2022-06-03 12:57:00', 'testiad'), ('SQL Server DB manual test 3.1.1.1B', '2022-06-03 13:57:00', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "3 rows affected" ja syötetyt rivit löytyvät tietokantataulusta.

3.1.1.1.3 Syötetään tauluun yksi rivi virheellisiä tietoja

Syötetään tauluun yksi rivi, jossa NoteDate –kentässä on päivämääärän sijaan tekstityyppistä tietoa.

```
Insert into Notes (NoteText, NoteDate, UserId) values ('SQL Server DB manual test 3.1.1.1C', 'Manual test', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Conversion failed when converting the date and/or time from character string." Syötettyä riviä ei löydy tietokantataulusta.

3.1.1.1.4 Syötetään tauluun rivi, josta puuttuu pakolliseksi määritellyjä tietoja

Syötetään tauluun rivi, jossa NoteText-kentästä puuttuu tieto.

```
Insert into Notes (NoteText, NoteDate, UserId) values (NULL, '22-06-03 10:57:00', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Cannot insert the value NULL into Column 'NoteText', table 'kipukalenteri.dbo.Notes'; column does not allow nulls. INSERT fails. The statement has been terminated" eikä syötettyä riviä löydy tietokannasta.

3.1.1.1.5 Muokataan tauluun syötettyjä tietoja

Muokataan kohdassa 3.1.1.1A syötettyä riviä vaihtamalla NoteText-kenttään teksti "SQL Server DB manual test 3.1.1.1E".

```
Update Notes Set NoteText = 'SQL Server DB manual test 3.1.1.1E' Where NoteID = 162
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty muutos löytyy oikealta tietokantariviltä.

3.1.1.1.6 Haetaan taulun kaikki tiedot

Haetaan tietokantataulun kaikki rivit.

```
Select * From Notes
```

Odotettu tulos: Taulun rivit näkyvät Results –osiossa.

3.1.1.1.7 Haetaan tietoja määritellyllä hakuehdolla

Haetaan Notes-taulun rivi, jonka NoteID vastaa testitapaus 3.1.1.1A-kohdassa syötetyn rivin id:tä

```
Select * From Notes Where NoteID = 162
```

Odotettu tulos: Kohdassa 3.1.1.1A syötetty rivi näkyy Results-osiossa.

3.1.1.1.8 Poistetaan taulusta rivi

Poistetaan yksi kohdassa 3.1.1.1B-syötetyistä riveistä tietokantataulusta päivämääräehdon perusteella.

```
Delete from Notes Where NoteDate = '2022-06-03 12:57:00'
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected", eikä poistettua riviä löydy tietokantataulusta.

3.1.1.2 PainLocation–taulun manuaalinen testaus

PainLocation–tauluun tallennetaan kipukohtien sijainnit. Taulussa on ala olevat kentät.

Pääavain LocationId tulee tietokannasta automaattisesti. Sillä on identity –ominaisuus, joten tietoa ei syötetä/muokata/poisteta taulusta.

LocationId	Sijainnin id	int	primary key	Not null
LocationName	Sijainnin nimi	nvarchar		Not null

3.1.1.2.1 Syötetään tauluun yksi rivi oikeellisia tietoja

Syötetään tauluun rivi, jossa on vähintään kaikki vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLocation(LocationName) values ('SQL Server DB manual test 1')
```

Odotettu tulos: SLQ Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty rivi löytyy tietokantataulusta.

3.1.1.2.2 Syötetään tauluun useita rivejä oikeellisia tietoja

Syötetään tauluun kolme riviä, joissa on vähintään vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLocation(LocationName) values
('SQL Server DB manual test 2'),
('SQL Server DB manual test 3'),
('SQL Server DB manual test 4')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "3 rows affected" ja syötetyt rivit löytyvät tietokantataulusta.

3.1.1.2.3 Syötetään tauluun rivi, josta puuttuu pakolliseksi määritellyjä tietoja

Syötetään tauluun rivi, jossa LocationName-kentästä puuttuu tieto.

```
Insert into PainLocation(LocationName) values (NULL)
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Cannot insert the value NULL into Column 'LocationName, table 'kipukalenteri.dbo.PainLocation'; column does not allow nulls. INSERT fails. The statement has been terminated" eikä syötettyjä tietoja löydy tietokannasta.

3.1.1.2.4 Muokataan tauluun syötettyjä tietoja

Muokataan kohdassa 3.1.1.2A syötettyä rivää vaihtamalla NoteText-kenttään teksti "SQL Server DB manual test 5".

```
Update PainLocation Set LocationName = 'SQL Server DB manual test 5' Where
LocationID = 3158
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty muutos löytyy oikealta tietokantariviltä.

3.1.1.2.5 Haetaan taulun kaikki tiedot

Haetaan tietokantataulun kaikki rivit.

```
Select * From PainLocation
```

Odotettu tulos: Taulun rivit näkyvät Results -osiossa.

3.1.1.2.6 Haetaan tietoja määritellyllä hakuehdolla

Haetaan taulun rivi, jonka LocationId vastaa testitapaus 3.1.1.2A-kohdassa syötetyn rivin päivämäärää.

```
Select * From PainLocation Where LocationID = 3158
```

Odotettu tulos: Kohdassa 3.1.1.2A syötetty rivi näkyy Results-osiossa.

3.1.1.2.7 Poistetaan taulusta rivi

Poistetaan yksi kohdassa 3.1.1.2B-syötetyistä riveistä tietokantataulusta.

```
Delete from PainLocation Where LocationID = 3160
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja poistettu rivi puuttuu tietokantataulusta.

3.1.1.3 PainLog-taulun manuaalinen testaus

PainLog-tauluun tallennetaan kipumerkinnät. Taulussa on alla olevat kentät. Pääavain LogId tulee tietokannasta automaattisesti. Sillä on identity -ominaisuus, joten tietoa ei syötetä/muokata/poisteta taulusta. UserId tulee tauluun automaattisesti käyttäjän käyttöliittymästä tekemän tiedon lisäyksen kautta. Taulussa on trigger, joka tuottaa tiedon syöttämisen tai muokkaamisen jälkeen Duration-kenttään tiedon EndTime-sarakkeen ja StartTime-sarakkeen erotuksena. Triggerin testaus on käsitelty erikseen.

LogId	Merkinnän id	int	primary key	Not null
PainIntensity	Kivun intensiteetti	int		
StartTime	Päivämäärä	datetime		Not null
EndTime	Päivämäärä	datetime		Not null
Duration	Kesto	int		
PainTrigger	Aiheuttaja	nvarchar		
PainType	Kivun tyyppi	nvarchar		
LocationId	Sijainnin id	int	foreign key(PainLocations-Locationid)	Not null
LocationInfo	Sijainnin lisätieto	nvarchar		
Medication	Lääkitys	nvarchar		
UserId	Käyttäjän id	nvarchar	foreign key(Users-Username)	Not null
Notes	Lisätiedot	nvarchar		

3.1.1.3.1 Syötetään tauluun yksi rivi oikeellisia tietoja

Syötetään tauluun rivi, jossa on vähintään kaikki vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserID)
values
(6, '2022-06-02 11:45:00', '2022-06-03 12:45:00', 107, 'SQL Server manual test
3.1.1.3A', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "LogId-table after insert trigger Tr_CalculateDurationMinutes DONE" ja "1 row affected". Syötetty rivi löytyy tietokantataulusta.

3.1.1.3.2 Syötetään tauluun useita rivejä oikeellisia tietoja

Syötetään tauluun kolme riviä, joissa on vähintään vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserID)
values
(3, '2022-06-01 11:30:00', '2022-06-03 18:00:00', 107, 'SQL Server manual test
3.1.1.3B', 'testiad'),
(4, '2022-06-01 10:30:00', '2022-06-02 11:30:00', 108, 'SQL Server manual test
3.1.1.3B', 'testiad'),
(5, '2022-06-01 08:55:00', '2022-06-01 09:45:00', 108, 'SQL Server manual test
3.1.1.3B', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen ilmoituksen "LogId-table after insert trigger Tr_CalculateDurationMinutes DONE" ja "3 rows affected" ja syötetyt rivit löytyvät tietokantataulusta.

3.1.1.3.3 Syötetään tauluun yksi rivi virheellisiä tietoja

Syötetään tauluun yksi rivi, jossa LocationId –kentässä on numeron sijaan tekstityyppistä tietoa.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserID)
values
(3, '2022-06-01 11:30:00', '2022-06-03 18:00:00', 'SQL', 'SQL Server manual test
3.1.1.3B', 'testiad')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Conversion failed when converting the varchar value 'SQL' to data type int. Riviä ei löydy tietokantataulusta.

3.1.1.3.4 Syötetään tauluun rivi, josta puuttuu pakolliseksi määritellyjä tietoja

Syötetään tauluun rivi, jossa UserID-kentästä puuttuu tieto.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserID)
values
(3, '2022-06-01 11:30:00', '2022-06-03 18:00:00', 107, 'SQL Server manual test
3.1.1.3B', NULL)
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Cannot insert the value NULL into Column 'UserID', table 'kipukalenteri.dbo.PainLogs'; column does not allow nulls. INSERT fails. The statement has been terminated" eikä syötettyjä tietoja löydy tietokannasta.

3.1.1.3.5 Muokataan tauluun syötettyjä tietoja

Muokataan kohdassa 3.1.1.3A syötettyä riviä vaihtamalla Notes-kenttään teksti "DB manual test 3.1.1.3E".

```
Update PainLog Set Notes = 'SQL Server DB manual test 3.1.1.3E' Where logID = 4054
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "LogId-table after update trigger Tr_CalculateDurationMinutes DONE" ja "LogId-table after insert trigger Tr_CalculateDurationMinutes DONE" ja "1 row affected" ja syötetty muutos löytyy oikealta tietokantariviltä.

3.1.1.3.6 Haetaan taulun kaikki tiedot

Haetaan tietokantataulun kaikki rivit.

```
Select * From PainLog
```

Odotettu tulos: Taulun rivit näkyvät Results -osiossa.

3.1.1.3.7 Haetaan tietoja määritellyllä hakuehdolla

Haetaan taulun rivit, joiden LocationID-kentässä on arvo 107

```
Select * From PainLog Where LocationID = 107
```

Odotettu tulos: Results-osiassa näkyvät rivit, joiden LocationID on 107.

3.1.1.3.8 Poistetaan taulusta rivi

Poistetaan yksi kohdassa 3.1.1.3B-syötetyistä riveistä tietokantataulusta hakuehtona LocationID:stä ja Durationia.

```
Delete from PainLog Where LocationID = 108 and Duration = 1500
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" eikä poistettua riviä löydy tietokantataulusta.

3.1.1.4 Users-taulun manuaalinen testaus

Users -tauluun tallennetaan käyttäjätiedot. Taulussa on alla olevat kentät. Käyttäjätunnus username toimii taulun pääavaimena.

Username	Käyttäjänimi	nvarchar	primary key	Not null
Email	Sähköposti	nvarchar		
Password	Salasana	nvarchar		
Token				

3.1.1.4.1 Syötetään tauluun yksi rivi oikeellisia tietoja

Syötetään tauluun rivi, jossa on vähintään kaikki vaaditut tiedot oikeassa tietotyypissä.

```
Insert into Users (Username, Email, Password) values
('Testi2', 'testi@testi', 'ManualTest1')
```

Odotettu tulos: SLQ Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty rivi löytyy tietokantataulusta.

3.1.1.4.2 Syötetään tauluun useita rivejä oikeellisia tietoja

Syötetään tauluun kolme riviä, joissa on vähintään vaaditut tiedot oikeassa tietotyypissä.

```
Insert into Users (Username, Email, Password) values
('Testi3', 'testi@testi', 'ManualTest1'),
('Testi4', 'testi@testi', 'ManualTest2'),
('Testi5', 'testi@testi', 'ManualTest3')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "3 rows affected" ja syötetyt rivit löytyvät tietokantataulusta.

3.1.1.4.3 Syötetään tauluun rivi, josta puuttuu pakolliseksi määritellyjä tietoja

Syötetään tauluun rivi, jossa Username-kentästä puuttuu tieto.

```
Insert into Users (Username, Email, Password) values  
(NULL, 'testi@testi', 'ManualTest1')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "Cannot insert the value NULL into Column 'Username', table 'kipukalenteri.dbo.Users; column does not allow nulls. INSERT fails. The statement has been terminated" eikä syötettyjä tietoja löydy tietokannasta.

3.1.1.4.4 Muokataan tauluun syötettyjä tietoja

Muokataan kohdassa 3.1.1.4A syötettyä riviä vaihtamalla Username-kenttään teksti "ManualTest4D"

```
Update Users set Username = 'ManualTest4D' Where Username = 'Testi2'
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja syötetty muutos löytyy oikealta tietokantariviltä.

3.1.1.4.5 Haetaan taulun kaikki tiedot

Haetaan tietokantataulun kaikki rivit.

```
Select * From Users
```

Odotettu tulos: Taulun rivit näkyvät Results -osiossa.

3.1.1.4.6 Haetaan tietoja määritellyllä hakuehdolla

Haetaan taulun rivi, jonka käyttäjänimi vastaa testitapaus 3.1.1.4D-kohdassa muokattua riviä.

```
Select * From Users Where Username = 'ManualTest4D'
```

Odotettu tulos: Kohdassa 3.1.1.4D syötetty rivi näkyy Results-osiolla.

3.1.1.4.7 Poistetaan taulusta rivi

Poistetaan yksi kohdassa 3.1.1.4B-syötetyistä riveistä tietokantataulusta.

```
Delete from Users Where Username = 'Testi4'
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "1 row affected" ja poistettu rivi puuttuu tietokantataulusta.

3.1.1.5 Tietokannan triggerin manuaalinen testaus

PainLog-tietokantataulussa on trigger, jonka tehtäväänä on tiedon lisäämisen tai päivittämisen jälkeen laskea Duration-kenttään kivun kesto EndTime-kentän ja StartTime-kentän arvojen erotuksena. Triggerin luontilause löytyy suunnittelukuvastosta tietokannan luontilauseiden yhteydestä.

Testausvaiheessa on olennaista testata triggerin toimivuus INSERT- ja UPDATE -lauseiden osalta lisättäessä tai muokattaessa yksittäisiä tai useita rivejä tietoa. DELETE-lauseen osalta testausta ei ole tarpeen tehdä, koska poistetulle riville muokkausta ei ole tarpeen tehdä.

Käyttöliittymässä ei ole mahdollisuutta lisätä tai muokata tietoa useita rivejä kerrallaan, mutta triggerissä on huomioitu tämä mahdollisuus siltä varalta että tietoja täytyy joskus lisätä useita rivejä kerrallaan suoraan tietokannan hallintaohjelman kautta.

3.1.1.5.1 Syötetään PainLog-tauluun yksi rivi tietoja.

Syötetään tauluun rivi, jossa on vähintään kaikki vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserId)
values
(6, '2022-06-03 19:07:00', '2022-06-03 20:35:00', 107, 'SQL Server manual test
trigger 1', 'anna1')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen "PainLog-table after insert trigger Tr-CalculateDurationMinutes DONE" ja "1 row affected". Syötetty rivi löytyy tietokantataulusta ja kivun kesto on laskettu oikein.

3.1.1.5.2 Syötetään tauluun useita rivejä oikeellisia tietoja

Syötetään tauluun neljä riviä, joissa on vähintään vaaditut tiedot oikeassa tietotyypissä.

```
Insert into PainLog (PainIntensity, StartTime, EndTime, LocationID, Notes, UserId)
values
(6, '2022-06-02 10:05:00', '2022-06-03 10:35:00', 107, 'SQL Server manual test
trigger 2', 'anna1'),
(6, '2022-06-01 19:05:00', '2022-06-01 19:06:00', 108, 'SQL Server manual test
trigger 3', 'Esa1'),
(6, '2022-06-04 11:35:00', '2022-06-04 11:42:00', 105, 'SQL Server manual test
trigger 4', 'Testi1'),
(6, '2022-06-05 14:57:00', '2022-06-05 15:07:00', 108, 'SQL Server manual test
trigger 5', 'Esa1')
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen triggerin käytöstä ja ilmoituksen "4 rows affected" ja ilmoituksen triggerin käytöstä. Syötetyt rivit löytyvät tietokantataulusta ja kivun kestot on laskettu oikein.

3.1.1.5.3 Muokataan PainLogs-taulun yhden rivin päivämäärätietoa

Muokataan yhden kohdassa 3.1.1.5B syötetyn rivin alkupäivämäärää 10 minuuttia aiemmaksi.

```
Update PainLog set StartTime = '2022-06-05 14:47:00' Where logID = 4065
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen triggerin käytöstä ja ilmoituksen "1 row affected". Muokattu rivi löytyy tietokantataulusta ja kivun kesto on laskettu oikein.

3.1.1.5.4 Muokataan PainLogs-taulun useamman rivin tietoja

Muokataan niiden taulun rivien StartTime-tietoa, jossa LocationID on 108.

```
Update PainLog set StartTime = '2022-06-01 08:30:00' Where LocationID = 108
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen triggerin käytöstä ja "3 rows affected". Muokatut rivit löytyvät tietokantataulusta ja kivun kestot on laskettu oikein.

3.1.1.5.5 Muokataan StartTime myöhäisemmäksi kuin EndTime

Muokataan yhden rivin StartTime eli alkuaika myöhäisemmäksi kuin EndTime

```
Update PainLog set StartTime = '2022-06-04 11:47:00' Where logID = 4064
```

Odotettu tulos: SQL Server Management Studio antaa ilmoituksen triggerin käytöstä ja "1 row affected". Muokatut rivit löytyvät tietokantataulusta ja kivun kesto on laskettu oikein.

3.1.1.6 Tietokantataulujen yhdistelyhaut

Tietokantataulujen tietoja yhdistellään käyttöliittymässä Kipumerkinnät-näkymässä. Siksi testataan myös tietokantataulujen yhdistelyhaut Join -liitoksilla.

3.1.1.6.1 Haetaan PainLog-taulun rivejä, joihin yhdistetään Users-taulun tietoja

Haetaan PainLog-taulun kenttiä tietyltä käyttäjältä ja yhdistetään hakuun käyttäjää vastaavat Username ja Email -kentät Users-taulusta.

```
Select p.logID, p.PainIntensity, p.StartTime, p.Duration, p.LocationID,
u.Username, u.Email
from PainLog as p Join Users as u on p.UserId = u.Username Where p.UserId = 'Esa1'
```

Odotettu tulos: Haetut kentät löytyvät Results-osiosta

3.1.1.6.2 Haetaan PainLog-taulun rivejä, joihin yhdistetään PainLocation-taulun tietoja

Haetaan PainLog-taulun kenttiä tietyltä käyttäjältä ja yhdistetään hakuun LocationID:tä vastaava sijainnin nimi PainLocation-taulusta:

```
Select p.logID, p.PainIntensity, p.StartTime, p.Duration, p.LocationID,
l.LocationName, p.UserId
from PainLog as p Join PainLocation as l on p.LocationID = l.LocationID Where
p.UserId = 'Esa1'
```

Odotettu tulos: Haetut rivit löytyvät Results osiosta.

3.1.1.6.3 Haetaan PainLog-taulun rivejä, joihin yhdistetään PainLocation-taulun tietoja

Haetaan PainLog-taulun kenttiä tietyltä käyttäjältä, yhdistetään hakuun sekä LocationID:tä vastaava LocationName PainLocation -taulusta sekä Username ja Email Users -taulusta.

```
Select p.logID, p.PainIntensity, p.StartTime, p.Duration, l.LocationName,
u.Username, u.Email
from PainLog as p Join PainLocation as l on p.LocationID = l.LocationID Join Users
as u on p.UserId = u.Username
Where p.UserId = 'Esa1'
```

Odotettu tulos: Haetut tiedot löytyvät Results osiosta.

3.1.2 Kontrollerien manuaalinen testaus Postman-sovelluksella

ASP.NET Core API-kontrollerit testataan manuaalisesti Postman-sovelluksella `HttpGet`-, `HttpPost`, `HttpPut` ja `HttpDelete` -pyynnöillä. Sovelluksen avulla lisätään oikeellista, virheellistä ja puutteellista testidataa kontrollerien kautta tietokantatauluihin, suoritetaan tietokantahakuja sekä muokataan ja poistetaan testidataa. Testattavana ovat Note-, PainLog, PainLocation- ja UserControllerien metodit `GetAll()`, `GetOne()`, `CreateNew()`, `Update()` ja `Delete()`.

3.1.2.1 Note-kontrollerin manuaalinen testaus

Note-kontrollerin tehtävä on välittää tietoa tietokannan Notes-taulun ja käyttöliittymän Muistiinpanot –näkymän toimintojen välillä. Kontrollerissa on alla olevat metodit:

GetAll()	<code>HttpGet</code> , hakee kaikki taulun tiedot
GetOne()	<code>HttpGet</code> , hakee yhden rivin tiedot id:llä
CreateNew()	<code>HttpPost</code> , luo uuden rivin
Update()	<code>HttpPut</code> , muokkaa rivin tietoja
Delete()	<code>HttpDelete</code> , poistaa rivin taulusta

3.1.2.1.1 Haetaan taulun kaikki tiedot

Haetaan taulun tiedot `HttpGet`-metodilla (kontrollerissa `GetAll()`-metodi).

GET https://localhost:5001/kipukalenteri/note

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja taulun tiedot palautuvat Body-osiassa.

3.1.2.1.2 Haetaan yksi rivi tietoa

Haetaan taulusta yksi rivi tietoa `HttpGet`-metodilla (kontrollerissa `GetOne()`-metodi).

GET https://localhost:5001/kipukalenteri/note/162

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja rivin tiedot näkyvät Body-osiassa.

3.1.2.1.3 Lisätään yksi rivi tietoa

Lisätään tietoa `HttpPost`-metodilla (kontrollerissa `CreateNew()`-metodi)

POST https://localhost:5001/kipukalenteri/note

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {
2 "noteText": "API manual test 3.1.2.1C",
3 "noteDate": "2022-06-04T10:57:00",
4 "userId": "anna1"
5 }

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiassa on ilmoitus "Muistiinpano X lisätty".

3.1.2.1.4 Muokataan tietokantaan syötettyä tietoa

Muokataan kohdassa 3.1.2.1A syötettyä riviä lisäämällä NoteText-kenttään teksti "Controller manual test via Postman 3.1.2.1D".

```

1 ...
2 ... "noteId": 166,
3 ... "noteText": "API manual test 3.1.2.1D",
4 ... "noteDate": "2022-06-04T10:57:00",
5 ... "userId": "anna1"
6 ...

```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiolla palautuu ilmoitus "Muistiinpano XXX päivitetty". Päivitetty tieto löytyy kannasta HttpGet-metodilla.

3.1.2.1.5 Poistetaan tietokantaan syötettyä tietoa

Poistetaan yksi kohdassa 3.1.1.1B syötetyistä riveistä tietokannasta.

KEY	VALUE
Key	Value

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiolla palautuu ilmoitus "Muistiinpano 165 poistettiin". Poistettua riviä ei löydy HttpGet-metodilla.

3.1.2.1.6 Syötetään rivi, joka sisältää väärän typpistä tietoa

Syötetään HttpPost-metodilla rivi, jossa noteDate-kenttään yritetään syöttää testitietoa.

```

1 ...
2 ... "noteText": "API manual test 3.1.2.1D",
3 ... "noteDate": "abcd",
4 ... "userId": "anna1"
5 ...

```

Odotettu tulos: Tiedon syöttäminen ei onnistu, Postman antaa statuksen 400 Bad Request.

3.1.2.1.7 Haetaan rivi tietoa väärän tyypisellä id:llä

Haetaan HttpGet-metodilla tietokantataulusta rivi id:llä, joka on tekstityyppinen.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Odotettu tulos: Haun status on 400 Bad Request ja Postman antaa ilmoituksen, että hakuarvo ei ole kelpaava.

3.1.2.1.8 Poistetaan rivi, jota ei ole olemassa

Poistetaan rivi id:llä jota ei löydy tietokantataulusta.

Odotettu tulos: Pyynnön status on 404 Not Found.

3.1.2.2 PainLocation-kontrollerin manuaalinen testaus

PainLocation-kontrollerin tehtävä on välittää tietoa tietokannan PainLocations-taulun ja käyttöliittymän Kivun sijainnit –näkymän toimintojen välillä. Kontrollerissa on alla olevat metodit:

GetAll()	HttpGet, hakee kaikki taulun tiedot
GetOne()	HttpGet, hakee yhden rivin tiedot id:llä
CreateNew()	HttpPost, luo uuden rivin
Update()	HttpPut, muokkaa rivin tietoja
Delete()	HttpDelete, poistaa rivin taulusta

3.1.2.2.1 Haetaan taulun kaikki tiedot

Haetaan taulun tiedot `HttpGet`-metodilla (kontrollerissa `GetAll()`-metodi).

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja taulun tiedot palautuvat Body-osiolla.

3.1.2.2.2 Haetaan yksi rivi tietoa

Haetaan taulusta yksi rivi tietoa `HttpGet`-metodilla (kontrollerissa `GetOne()`-metodi).

GET https://localhost:5001/kipukalenteri/painlocation/107

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja rivin tiedot näkyvät Body-osiolla.

3.1.2.2.3 Lisätään yksi rivi tietoa

Lisätään tietoa HttpPost-metodilla (kontrollerissa CreateNew()-metodi).

POST https://localhost:5001/kipukalenteri/painlocation

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 ...
2 ... "locationName": "Postman API test 1"
3 ...
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiolla ilmoituksessa palautuu lisätyn sijainnin nimi. Lisättyt tiedot löytyvät HttpGet-metodilla tietokannasta.

3.1.2.2.4 Muokataan tietokantaan syötettyä tietoa

Muokataan aiemmin syötettyä rivää lisäämällä LocationName-kenttään teksti "Postman API test 2"

PUT https://localhost:5001/kipukalenteri/painlocation/3162

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 ...
2 ... {
3 ...   "locationId": 3162,
4 ...   "locationName": "Postman API test 2"
5 ... }
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiolla palautuu ilmoitus sijainnin päivittämisestä. Muokattu rivi löytyy HttpGet-metodilla

3.1.2.2.5 Poistetaan tietokantaan syötettyä tietoa

Poistetaan yksi aiemmin syötetyistä riveistä tietokannasta.

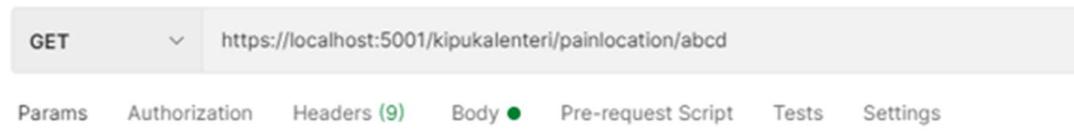
DELETE https://localhost:5001/kipukalenteri/painlocation/3159

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Odottettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" Body-osiolla palautuu ilmoitus sijainnin poistamisesta.

3.1.2.2.6 Haetaan tietoa vääräntyyppisellä id:llä

Haetaan tietoa tekstityyppisellä id:llä `HttpGet`-metodilla.



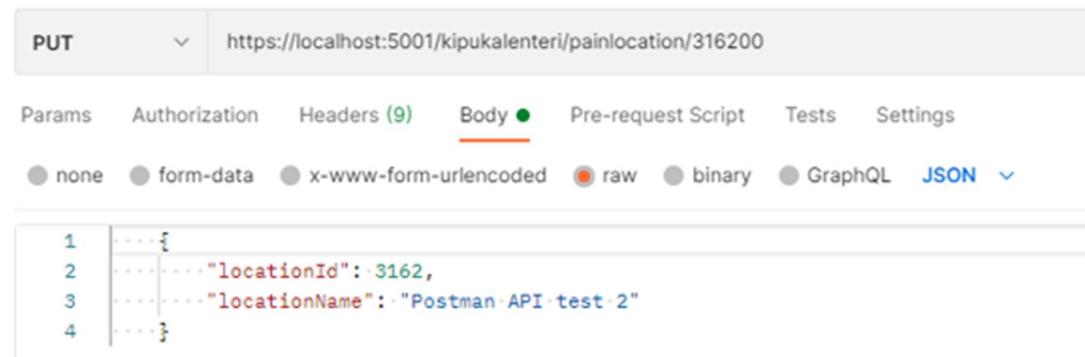
A screenshot of the Postman interface showing a GET request. The method is set to `GET`, the URL is `https://localhost:5001/kipukalenteri/painlocation/abcd`, and the Headers tab shows 9 items. The Body tab is selected, showing the raw JSON body:

```
1  ...
2  ...
3  ...
4  ...{  
5     "locationId": 3162,  
6     "locationName": "Postman API test 2"  
7 }
```

Odottettu tulos: Pyynnön status on 400 Bad Request, haku ei onnistu.

3.1.2.2.7 Muokataan riviä, jota ei ole olemassa

Laitetaan muokkauspyyntö `HttpPut`-metodilla `locationId`:lle, jota ei ole tietokantataulussa.



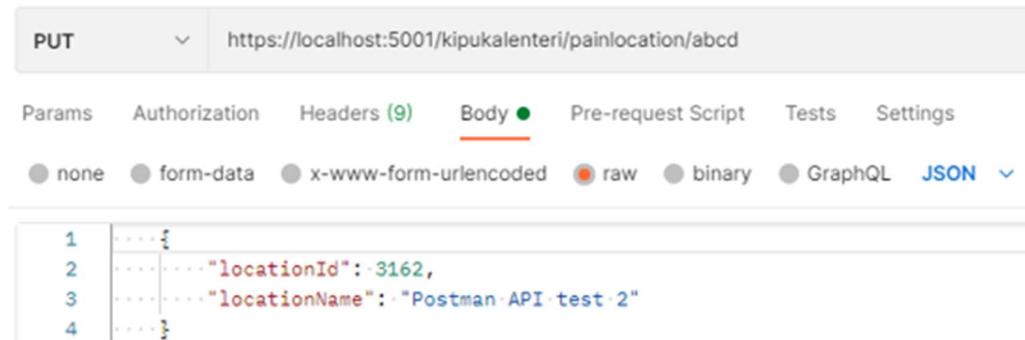
A screenshot of the Postman interface showing a PUT request. The method is set to `PUT`, the URL is `https://localhost:5001/kipukalenteri/painlocation/316200`, and the Headers tab shows 9 items. The Body tab is selected, showing the raw JSON body:

```
1  ...
2  ...
3  ...
4  ...{  
5     "locationId": 3162,  
6     "locationName": "Postman API test 2"  
7 }
```

Odottettu tulos: Pyynnön status on 404 Not Found ja Body-osiolla palautuu ilmoitus, että päivitytävää sijaintia ei löytynyt.

3.1.2.2.8 Muokataan riviä, jolle annetaan vääräntyyppinen id

Laitetaan muokkauspyyntö `HttpPut`-metodilla `locationId`:lle, joka on tekstityyppinen.



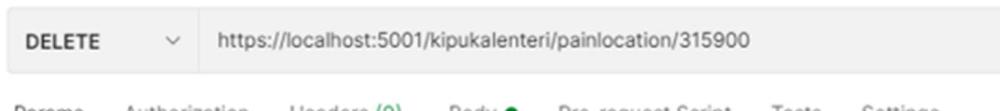
A screenshot of the Postman interface showing a PUT request. The method is set to `PUT`, the URL is `https://localhost:5001/kipukalenteri/painlocation/abcd`, and the Headers tab shows 9 items. The Body tab is selected, showing the raw JSON body:

```
1  ...
2  ...
3  ...
4  ...{  
5     "locationId": 3162,  
6     "locationName": "Postman API test 2"  
7 }
```

Odottettu tulos: Pyynnön status on 400 Bad Request, muokkaus ei onnistu.

3.1.2.2.9 Poistetaan rivi, jota ei löydy tietokantataulusta

Laitetaan `HttpDelete` -poistopyyntö `id`:lle, jota ei ole tietokantataulussa.



A screenshot of the Postman application interface. The method dropdown shows "DELETE". The URL field contains "https://localhost:5001/kipukalenteri/painlocation/315900". Below the URL, there are tabs for "Params", "Authorization", "Headers (9)", "Body ●", "Pre-request Script", "Tests", and "Settings". The "Headers" tab is currently selected.

Odotettu tulos: Pyynnön status on 404 Not Found ja Body-osiolla palautuu ilmoitus siitä, että sijaintia ei löydy. Poisto ei onnistu.

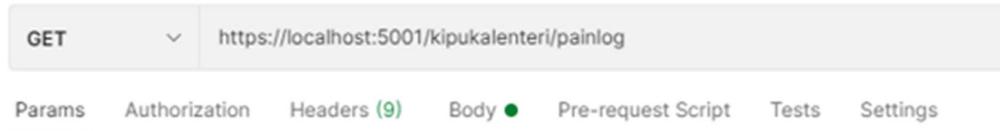
3.1.2.3 PainLog-kontrollerin manuaalinen testaus

PainLog-kontrollerin tehtävä on välittää tietoa tietokannan PainLog-taulun ja käyttöliittymän Kipumerkinnät –näkymän toimintojen välillä. Kontrollerissa on alla olevat metodit:

GetAll()	HttpGet, hakee kaikki taulun tiedot
GetOne()	HttpGet, hakee yhden rivin tiedot id:llä
CreateNew()	HttpPost, luo uuden rivin
Update()	HttpPut, muokkaa rivin tietoja
Delete()	HttpDelete, poistaa rivin taulusta

3.1.2.3.1 Haetaan taulun kaikki tiedot

Haetaan taulun tiedot `HttpGet`-metodilla (kontrollerissa `GetAll()`-metodi).

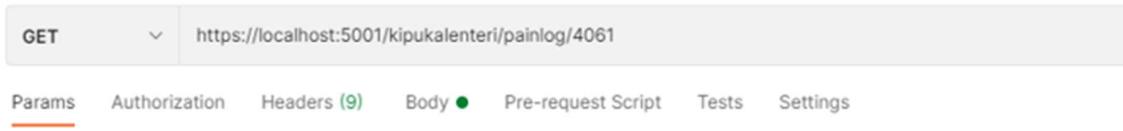


A screenshot of the Postman application interface. The method dropdown shows "GET". The URL field contains "https://localhost:5001/kipukalenteri/painlog". Below the URL, there are tabs for "Params", "Authorization", "Headers (9)", "Body ●", "Pre-request Script", "Tests", and "Settings". The "Headers" tab is currently selected.

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja taulun tiedot palautuvat Body-osiolla.

3.1.2.3.2 Haetaan yksi rivi tietoa

Haetaan taulusta yksi rivi tietoa `HttpGet`-metodilla (kontrollerissa `GetOne()`-metodi).



A screenshot of the Postman application interface. The method dropdown shows "GET". The URL field contains "https://localhost:5001/kipukalenteri/painlog/4061". Below the URL, there are tabs for "Params", "Authorization", "Headers (9)", "Body ●", "Pre-request Script", "Tests", and "Settings". The "Headers" tab is currently selected.

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja rivin tiedot näkyvät Body-osiolla.

3.1.2.3.3 Lisätään yksi rivi tietoa

Lisätään tietoa `HttpPost`-metodilla (kontrollerissa `CreateNew()`-metodi).

The screenshot shows a Postman request configuration. The method is POST, the URL is <https://localhost:5001/kipukalenteri/painlog>, and the Body tab is selected. The JSON input is:

```
1
2   ...
3     "painIntensity": 5,
4     "startTime": "2022-06-04T19:33:00",
5     "endTime": "2022-06-04T20:35:00",
6     "medication": null,
7     "locationInfo": null,
8     "painTrigger": null,
9     "painType": null,
10    "locationId": 105,
11    "notes": "Postman API manual test 1",
12    "userId": "testiad"
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osion ilmoituksessa palautuu lisätyn sijainnin nimi. Lisätty tiedot löytyvät HttpGet-metodilla tietokannasta.

3.1.2.3.4 Muokataan tietokantaan syötettyä tietoa

Muokataan aiemmin syötettyä riviä lisäämällä notes-kenttään teksti "Postman API manual test 2"

The screenshot shows a Postman request configuration. The method is PUT, the URL is <https://localhost:5001/kipukalenteri/painlog/4066>, and the Body tab is selected. The JSON input is:

```
1
2   ...
3     "logId": 4066,
4     "painIntensity": 7,
5     "startTime": "2022-06-04T19:33:00",
6     "endTime": "2022-06-04T20:35:00",
7     "medication": null,
8     "locationInfo": null,
9     "painTrigger": null,
10    "painType": null,
11    "locationId": 105,
12    "notes": "Postman API manual test 2",
13    "userId": "testiad"
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiassa palautuu ilmoitus merkinnän päivittämisestä. Muokattu rivi löytyy HttpGet-metodilla.

3.1.2.3.5 Poistetaan tietokantaan syötettyä tietoa

Poistetaan yksi aiemmin syötetyistä riveistä tietokannasta.

The screenshot shows a Postman request configuration. The method is DELETE, the URL is <https://localhost:5001/kipukalenteri/painlog/4062>, and the Body tab is selected. The JSON input is empty.

Odottettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" Body-osiassa palautuu ilmoitus sijainnin poistamisesta. Poistettua riviä ei löydy tietokannasta.

3.1.2.3.6 Haetaan tietoa vääräntyyppisellä id:llä

Haetaan tietoa tekstityyppisellä id:llä `HttpGet`-metodilla.

GET https://localhost:5001/kipukalenteri/painlog/abcd

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Odottettu tulos: Pyynnön status on 400 Bad Request, haku ei onnistu.

3.1.2.3.7 Haetaan riviä, jota ei ole olemassa

Haetaan `HttpGet`-metodilla riviä id:llä, jota ei ole tietokantataulussa.

GET https://localhost:5001/kipukalenteri/painlog/406133

Odottettu tulos: Pyynnön status on 204 No Content eikä haku onnistu.

3.1.2.3.8 Syötetään rivi, josta puuttuu vaadittavia tietoja

Lisätään rivi, josta puuttuu tietokannassa pakolliseksi määritelty `StartTime`.

POST https://localhost:5001/kipukalenteri/painlog

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1
2   "painIntensity": 7,
3   "startTime": null,
4   "endTime": "2022-06-04T20:35:00",
5   "medication": null,
6   "locationInfo": null,
7   "painTrigger": null,
8   "painType": null,
9   "locationId": 105,
10  "notes": "Postman API manual test 1",
11  "userId": "testi@ad"
12

```

Odottettu tulos: Pyynnön status on 400 Bad Request ja Body-osiassa palautuu ilmoitus "Jokin meni pieleen merkintää lisättäessä". Lisäys ei onnistu.

3.1.2.4 User-kontrollerin manuaalinen testaus

User-kontrollerin tehtävä on välittää tietoa tietokannan `Users`-taulun ja käyttöliittymän Käyttäjät –näkymän toimintojen välillä. Kontrollerissa on alla olevat metodit:

GetAll()	HttpGet, hakee kaikki taulun tiedot
GetOne()	HttpGet, hakee yhden rivin tiedot id:llä
CreateNew()	HttpPost, luo uuden rivin
Update()	HttpPut, muokkaa rivin tietoja
Delete()	HttpDelete, poistaa rivin taulusta

3.1.2.4.1 Haetaan taulun kaikki tiedot

Haetaan taulun tiedot `HttpGet`-metodilla (kontrollerissa `GetAll()`-metodi).

The screenshot shows a Postman request configuration. The method is set to `GET`, and the URL is `https://localhost:5001/kipukalenteri/user`. The request is currently empty.

Odottettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja taulun tiedot palautuvat Body-osiolla.

3.1.2.4.2 Haetaan yksi rivi tietoa

Haetaan taulusta yksi rivi tietoa `HttpGet`-metodilla (kontrollerissa `GetOne()`-metodi).

The screenshot shows a Postman request configuration. The method is set to `GET`, and the URL is `https://localhost:5001/kipukalenteri/user/anna1`. The request is currently empty.

Odottettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja rivin tiedot näkyvät Body-osiolla.

Tulos: Testi toteutuu odotetun tuloksen mukaisesti. Pyynnön status on 200 OK ja haetun rivin tiedot palautuvat Body-osiolla.

3.1.2.4.3 Lisätään yksi rivi tietoa

Lisätään tietoa `HttpPost`-metodilla (kontrollerissa `CreateNew()`-metodi).

The screenshot shows a Postman request configuration. The method is set to `POST`, and the URL is `https://localhost:5001/kipukalenteri/user`. The `Body` tab is selected, showing a raw JSON payload:

```
1
2   ...
3     ...
4       ...
5         ...
```

The JSON body is defined as:

```
{"username": "Postman1", "email": "postman@testi", "password": "Postman1"}
```

Odottettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiolla ilmoituksessa palautuu lisätyn sijainnin nimi. Lisättyt tiedot löytyvät `HttpGet`-metodilla tietokannasta.

3.1.2.4.4 Muokataan tietokantaan syötettyä tietoa

Muokataan aiemmin syötetyn rivin email-kenttää.

```
1
2   "username": "Postman1",
3   "email": "postman2@testi",
4   "password": "Postman1"
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" ja Body-osiossa palautuu ilmoitus merkinnän päivittämisestä. Muokattu rivi löytyy HttpGet-metodilla.

3.1.2.4.5 Poistetaan tietokantaan syötettyä tietoa

Poistetaan yksi aiemmin syötetyistä riveistä tietokannasta.

```
DELETE https://localhost:5001/kipukalenteri/user/Testi5
```

Odotettu tulos: Postman antaa ilmoituksen "Status: 200 Ok" Body-osiossa palautuu ilmoitus sijainnin poistamisesta.

3.1.2.4.6 Syötetään rivi, jonka username:lla on jo rivi tietokannassa

Lisätään tietokantatauluun rivi, jossa annetulla username-kentän päävaimella löytyy jo tietokannasta rivi.

```
POST https://localhost:5001/kipukalenteri/user
```

Odotettu tulos: Pyynnön status on 400 Bad Request ja Body-osiossa palautuu ilmotus "Jokin meni pieleen käyttäjää lisättäessä". Lisäys ei onnistu.

3.1.2.4H Muokataan käyttäjänimeä

Laitetaan HttpPut-metodilla muokauspyyntö, jossa muokataan username-kenttää (pääavain).

```
PUT https://localhost:5001/kipukalenteri/user/Postman1
Params Authorization Headers (9) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 ...
2 ...
3 ...
4 ...
5 ...
{
  "username": "Postman2",
  "email": "postman2@testi",
  "password": "Postman1"
}
```

Odotettu tulos: Pyynnön status on 400 Bad Request, muokkaus ei onnistu.

Tulos: Testi toteutuu odotetun tuloksen mukaisesti.

3.1.2.4.7 Poistetaan rivi, jota ei löydy tietokantataulusta

Laitetaan HttpDelete-metodilla poistopyyntö päävaimelle, jota ei löydy tietokantataulusta.

```
DELETE https://localhost:5001/kipukalenteri/user/Testi555
Params Authorization Headers (7) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL
{
  "username": "Postman2",
  "email": "postman2@testi",
  "password": "Postman1"
}
```

Odotettu tulos: Pyynnön status on 404 Not Found, poistoa ei toteuteta. Body-osiossa palautuu ilmoitus siitä, että käyttäjää ei löytynyt.

3.1.3 Käyttöliittymän manuaalinen testaus

Käyttöliittymän toiminnot testataan manuaalisesti aina kunkin toiminnon valmistuttua. Testaus suoritetaan kokeilemalla, että kyseisen taulun tietojen lisäys, muokkaus ja poisto onnistuu manuaalisesti käyttöliittymän kautta, toimii oikein ja annetut ilmoitukset ovat oikeanlaiset. Testauksessa yritetään myös syöttää/muokata virheellistä ja puuttuvaa dataa käyttöliittymän kautta.

3.1.3.1 Muistiinpanot-näkymän manuaalinen testaus

Muistiinpanot-näkymässä lisätään, poistetaan ja muokataan muistiinpanoja. Muistiinpanot näkyvät laskevassa päivämääräjärjestyksessä, näkyvillä on päivämäärä ja muistiinpanon teksti. Käyttäjälle näkyvät vain hänen omat muistiinpanonsa, käyttäjätieto tallentuu muistiinpanoon automaatisesti.

3.1.3.1.1 Lisätään muistiinpano oikeilla tiedoilla

Lisätään muistiinpano, jonka päivämäärä on aiemmin lisättyjä muistiinpanoja tuoreempi. Annetaan kaikki tietokannan vaatimat tiedot (päivämäärä ja muistiinpanon teksti)

Päivämäärä
04.06.2022 13.44

WebApp Manual Test 1

Käyttäjä-ID: testiad

Odotettu tulos: Sovellus antaa ilmoituksen ”Lisätty uusi muistiinpano” ja lisätty muistiinpano tulee näkymän ylimmäiseksi.

3.1.3.1.2 Lisätään muistiinpano puutteellisilla tiedoilla

Lisätään muistiinpano ilman päivämäärää.

Päivämäärä
pp.kk.vvvv --.--

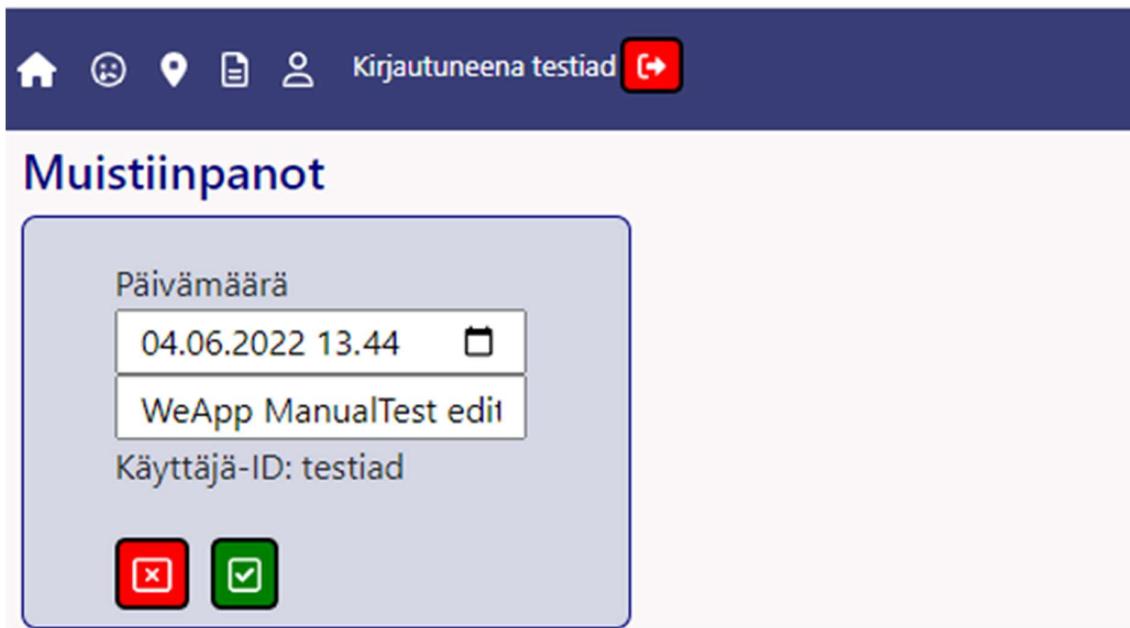
App Manual Test 2

Käyttäjä-ID: testiad

Odotettu tulos: Sovellus antaa virheilmoituksen eikä lisäys onnistu.

3.1.3.1.3 Muokataan olemassa olevaa muistiinpanoa

Muokataan aiemmin lisättyä muistiinpanoa muokkaamalla teksti-kenttää ”WebApp ManualTest edit”.



Odottettu tulos: Sovellus antaa ilmoituksen ”Päivitetty muistiinpanoa” ja muokattu teksti löytyy oikeasta muistiinpanosta.

3.1.3.1.4 Poistetaan muistiinpano

Poistetaan aiemmin lisätty muistiinpano.

Odottettu tulos: Sovellus varmistaa, halutaanko poisto tehdä ja vahvistuksen jälkeen antaa ilmoituksen ”Muistiinpanon poisto onnistui”. Poistettua merkintää ei löydy muistiinpanoista.

3.1.3.1.5 Tarkistetaan, että kaikki näkymän merkinnät ovat käyttäjän omia

Tarkistetaan, että kaikki näkymässä näkyvät merkinnät ovat myös tietokannassa merkitty kyseiselle käyttäjätunnuselle.

Odottettu tulos: Käyttäjän lisäämät muistiinpanot tietokannassa ja sovelluksessa vastaavat toisiaan.

3.1.3.2 Sijainnit-näkymän manuaalinen testaus

Sijainnit-näkymässä lisätään, poistetaan ja muokataan kivun sijainteja. Sijainnit näkyvät nousevassa Id-järjestysessä, näkyvillä on Id ja kivun sijainti. Käyttäjälle kaikki tietokannan sijainnit.

3.1.3.2.1 Lisätään sijainti oikeilla tiedoilla

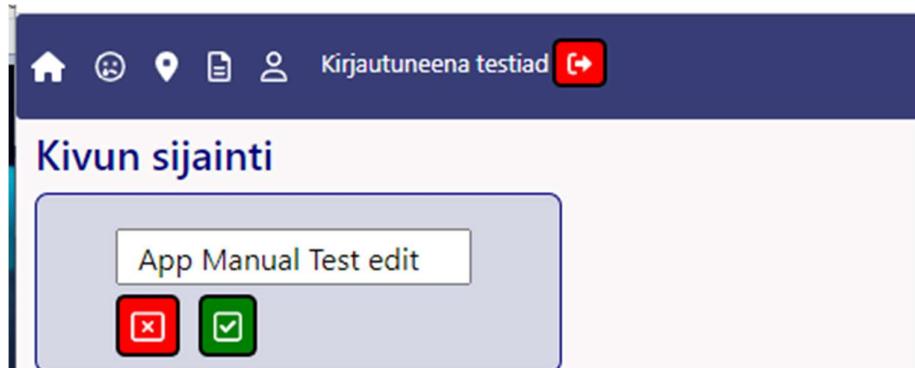
Lisätään sijainti.



Odotettu tulos: Sovellus antaa ilmoituksen "Lisätty uusi sijainti" ja lisätty sijainti löytyy listauksesta.

3.1.3.2.2 Muokataan olemassa olevaa sijaintia

Muokataan aiemmin lisättyä sijaintia muokkaamalla kenttää "App Manual Test edit".



Odotettu tulos: Sovellus antaa ilmoituksen "Päivitetty sijaintia App Manual Test edit" ja muokattu sijainti löytyy listauksesta.

3.1.3.2.3 Poistetaan sijainti

Poistetaan aiemmin lisätty sijainti.

Odotettu tulos: Sovellus varmistaa, halutaanko poisto tehdä ja vahvistuksen jälkeen antaa ilmoituksen "Sijainnin XXX poisto onnistui".

3.1.3.2.4 Tarkistetaan, että käyttäjälle näkyvät kaikki sijainnit

Tarkistetaan, että kaikki tietokantaan tallennetut sijainnit näkyvät käyttäjälle Sijainnit-listauksessa.

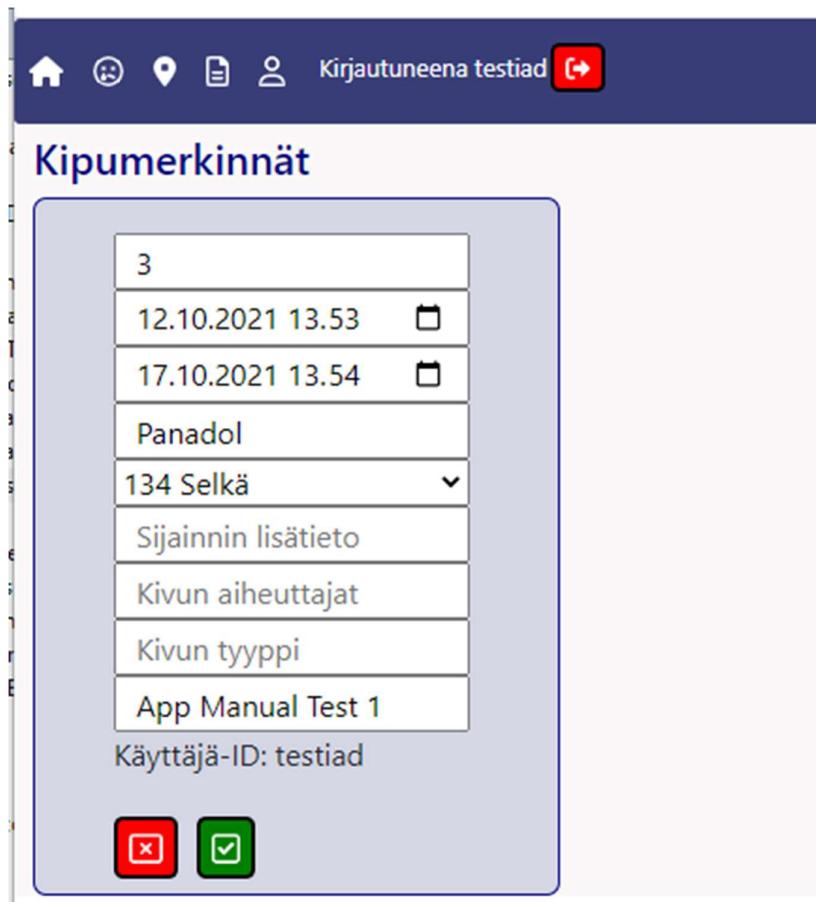
Odotettu tulos: Näkymän sijainnit ja tietokannan sijainnit vastaavat toisiaan.

3.1.3.3 Kipumerkinnät-näkymän manuaalinen testaus

Kipumerkinnät-näkymässä lisätään, poistetaan ja muokataan kipumerkintöjä. Merkinnät näkyvät nousevassa/laskevassa päivämääräjärjestysessä, näkyvillä on päivämäärä ja kivun sijainti. Käyttäjälle näkyvät vain hänen omat kipumerkintänsä, käyttäjätieto tallentuu merkintään automaatisesti.

3.1.3.3.1 Lisätään merkintä oikeilla tiedoilla

Lisätään kipumerkintä ja täytetään kaikki tietokannan vaativat kentät.



Odottettu tulos: Sovellus antaa ilmoituksen "Lisätty uusi merkintä" ja merkintä sijainti löytyy listauksesta. Kivun kesto tunneissa ja minuuteissa on laskettu oikein.

3.1.3.3.2 Muokataan olemassa olevaa merkintää

Muokataan aiemmin lisättyä merkintää muokkaamalla muistiinpanot-kenttää "App Manual Test edit".

Kipumerkinnät

Kivun alkamisaika
12.10.2021 13.53

Kivun loppumisaika
17.10.2021 13.54

Kivun intensiteetti 1 - 10
3

Lääkitys
Panadol

Kivun sijainti
134 Selkä

Sijainnin lisätieto

Kivun aiheuttajat

Kivun tyyppi

Lisätiedot
App Manual Test edit

Käyttäjä-ID: testiad

Odottettu tulos: Sovellus antaa ilmoituksen ”Päivitetty merkintää” ja muokattu merkintä löytyy listauksesta.

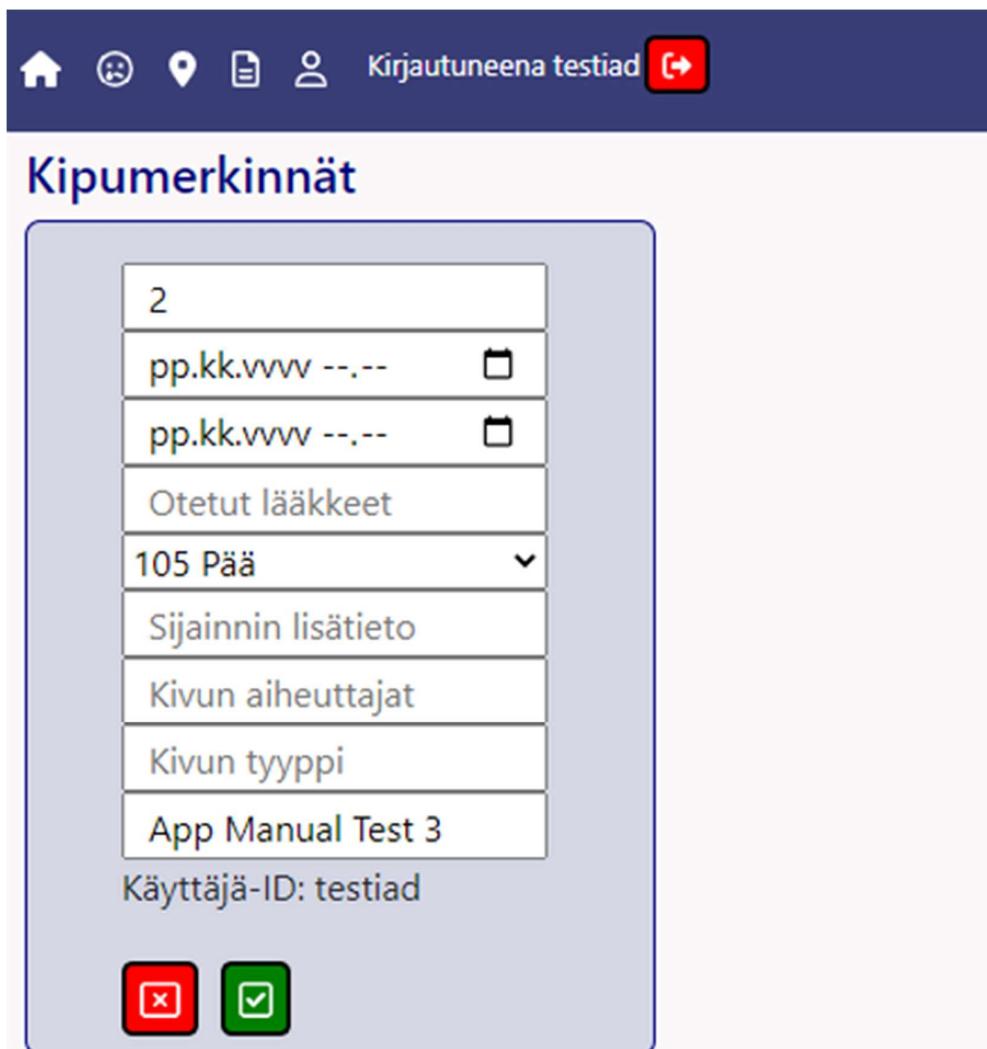
3.1.3.3.3 Poistetaan merkintä

Poistetaan aiemmin lisätty merkintä.

Odottettu tulos: Sovellus varmistaa, halutaanko poisto tehdä ja vahvistuksen jälkeen antaa ilmoituksen ”Merkinnän poisto onnistui”.

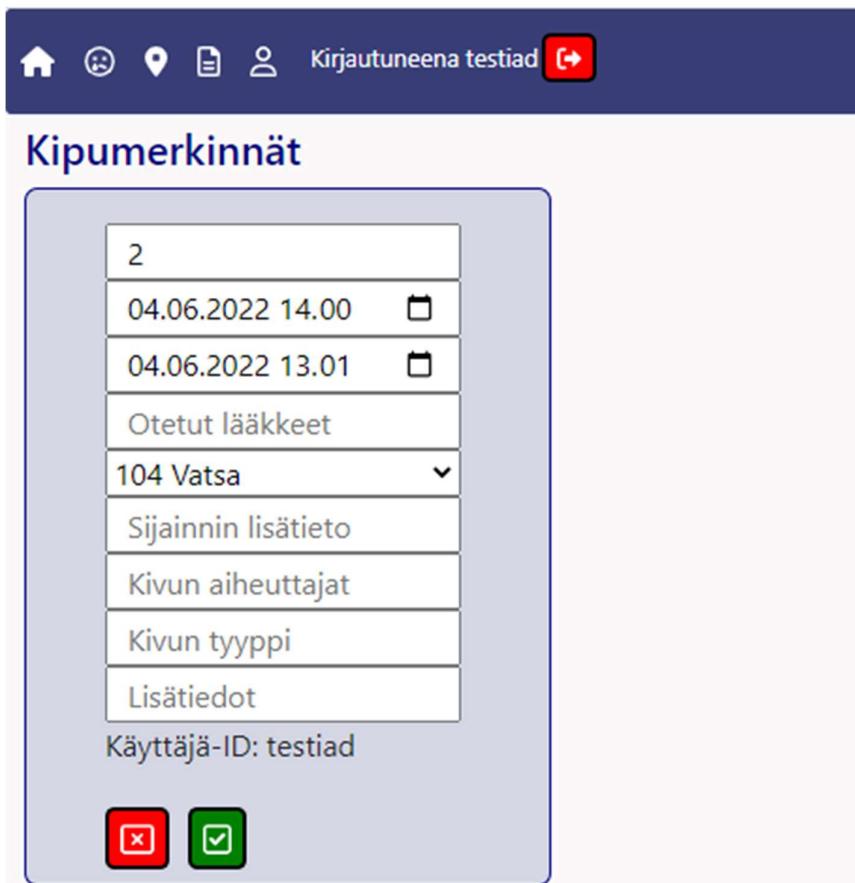
3.1.3.3.4 Lisätään merkintä puutteellisilla tiedoilla

Lisätään kipumerkintä ilman alku- ja päätymispäivää.



Odottettu tulos: Sovellus antaa virheilmoituksen, eikä lisäys onnistu.

- 3.1.3.3.5 Lisätään merkintä, jossa alkamisaika on myöhemmin kuin päättymisaika
Lisätään merkintä, jossa alkamisaika on myöhemmin kuin päättymisaika.



Odottettu tulos: Lisäys onnistuu, mutta kivun kesto on -59 minuuttia

3.1.3.3.6 Tarkistetaan, että käyttäjälle vain hänen kipumerkintänsä

Tarkistetaan, että käyttäjälle näkyvät vain hänen omat kipumerkintänsä.

Odottettu tulos: Näkymän kipumerkinnät vastaavat tietokannassa käyttäjälle merkittyjä kipumerkintöjä.

3.1.3.3.7 Kaikkien kipumerkintöjen kestot on oikein laskettu

Tarkistetaan, että kaikkien näkymän kipumerkintöjen kestot ovat oikein.

Odottettu tulos: Kaikkien kipumerkintöjen kestot on oikein laskettu.

Tulos: Kaikkien muiden kipumerkintöjen kestot on oikein laskettu, paitsi sen jossa kesto on negatiivinen eli kivun alkuaika myöhemmin kuin päätymisaika.

3.1.3.3.8 Sijaintisuodatin toimii

Testataan, että sijaintisuodatin suodattaa näkyviin oikeat merkinnät ja sijaintia voi suodattaa useita kertoja peräkkäin.

Odottettu tulos: Sijaintisuodattimen suodattamat merkinnät vastaavat tietokannan merkintöjä ja sijaintia voi suodattaa useita kertoja peräkkäin.

3.1.3.3.9 Aikasuodatin toimii

Testataan, että aikasuodatin suodattaa näkyviin oikeat merkinnät ja aikaa voi suodattaa useita kertoja peräkkäin.

Odotettu tulos: Aikasuodattimen suodattamat merkinnät vastaavat tietokannan merkintöjä ja merkintöjä voi suodattaa useita kertoja peräkkäin.

3.1.3.3.10 Molempia suodattimia voi käyttää yhtä aikaa useita kertoja peräkkäin

Testataan, että sekä aikasuodatinta että sijaintisuodatinta voi käyttää yhtä aikaa useita kertoja peräkkäin.

Odotettu tulos: Suodatuksit toimivat yhtä aikaa ja suodatusta voi muuttaa useita kertoja peräkkäin.

3.1.3.4 Käyttäjät-näkymän manuaalinen testaus

Käyttäjät-näkymässä lisätään, poistetaan ja muokataan käyttäjiä. Käyttäjälle näkyy vain hänen oma tunnuksensa, mutta hän voi luoda uusia tunnuksia.

3.1.3.4.1 Lisätään käyttäjä sovelluksen kautta

Lisätään käyttäjä käyttäjänimellä "AppManualTest" sovellukseen.

Odotettu tulos: Sovellus antaa ilmoituksen "Lisätty käyttäjä AppManualTest". Lisättyä käyttäjää ei näy näkymässä, mutta se löytyy tietokannasta.

3.1.3.4.2 Muokataan käyttäjän sähköpostia

Muokataan käyttäjän sähköpostiksi testi11@testi.fi

Käyttäjät

testi11@testi.fi
.....
.....

Käyttäjä: testiad

Odotettu tulos: Sovellus antaa ilmoituksen ”Päivitetty käyttäjää testiad” ja käyttäjän sähköposti vastaa päivitystä.

3.1.3.4.3 Poistetaan käyttäjä

Yritetään poistaa käyttäjä.

Odotettu tulos: Sovellus varmistaa, halutaanko poisto tehdä. Vahvistuksen jälkeen sovellus antaa virheilmoituksen, eikä poistoa tehdä. Käyttäjällä on tietokannassa muistiinpanoja ja kipumerkintöjä, eikä poistoa voi tehdä koska käyttäjänimi toimii viiteavaimena näissä merkinnöissä.

3.1.3.4.4 Tarkistetaan, että käyttäjälle näkyy vain hänen oma tunnuksensa

Tarkistetaan, että käyttäjälle näkyy vain hänen oma tunnuksensa.

Odotettu tulos: Käyttäjälle näkyy vain yksi käyttäjätunnus, hänen omansa.

3.1.3.5 Kirjautumisen manuaalinen testaus

3.1.3.5.1 Kirjautuminen tunnuksilla, joita ei ole olemassa

Yritetään kirjautua käyttäjätunnusella ja salasanalla, jota ei ole olemassa

admin
.....

Odotettu tulos: Kirjautuminen ei onnistu.

3.1.3.5.2 Käyttäjän lisääminen kirjautumatta

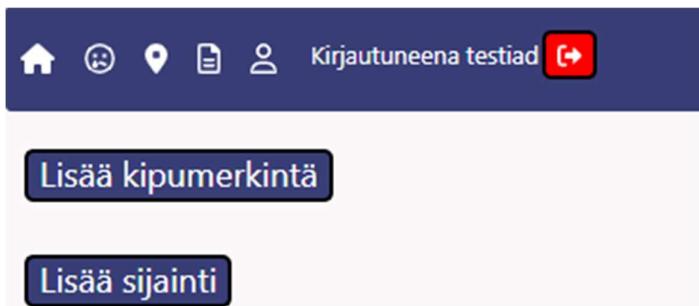
Lisätään käyttäjä Manual1 kirjautumatta ja kirjaudutaan sen jälkeen luodulla tunnuksilla sovellukseen.



Odotettu tulos: Lisäys onnistuu ja käyttäjä pääsee kirjautumaan sovellukseen.

3.1.3.5.3 Lisää kipumerkintä -nappi toimii

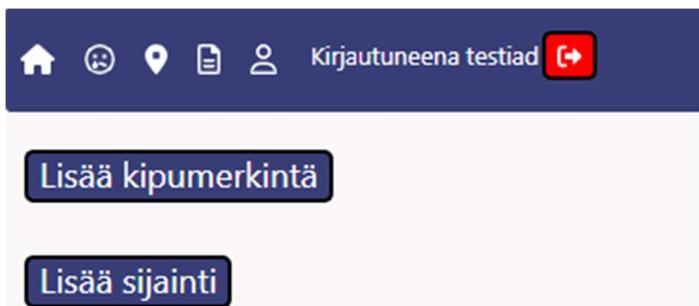
Testataan, että etusivun lisää kipumerkintä -nappi avaa Lisää kipumerkintä -komponentin ja kipumerkinnän lisäys toimii.



Odotettu tulos: Nappi avaa Lisää kipumerkintä -komponentin. Kipumerkinnän lisäys onnistuu, lisäyksen jälkeen sovellus palaa etusivulle.

3.1.3.5.4 Lisää sijainti -nappi toimii

Testataan, että etusivun lisää sijainti -nappi avaa Lisää sijainnin -komponentin ja sijainnin lisäys toimii.



Odottettu tulos: Nappi avaa Lisää sijainti -komponentin. Sijainnin lisäys onnistuu, lisäyksen jälkeen sovellus palaa etusivulle.

3.2 Automaattinen testaus

Kipukalenteri-sovelluksen automaattinen testaus suoritetaan kolmessa vaiheessa.

Yksikkötestaus suoritetaan React-sovelluksella Jest-testikirjastolla sovelluksen komponenttien valmistuttua. Integraatiotestaus toteutetaan .NET Core REST Api-sovelluksessa Alba- ja xunit -testikirjastoilla kontrollerien valmistumisen jälkeen. Lopuksi suoritetaan end-to-end -testaus Cypress-testikirjastolla sovelluksen valmistumisen jälkeen. Testauksen lopputulos on käsitelty tämän dokumentin liitteenä olevassa testauspöytäkirjassa.

3.2.1 Yksikkötestaus Jest-kirjastolla

Kipukalenteri-sovelluksen yksikkötestit suoritetaan front end React-sovelluksessa Jest-testikirjastolla. Testauksen kohteena ovat Note-, Location-, PainLog- ja Users-kansioista löytyvät komponenttit. Testauksessa käydään läpi nappien ja lomakkeiden toimintaa sekä sivun renderöintiä.

3.2.1.1 *Note-kansion komponenttien automaattinen testaus*

3.2.1.1.1 Note-komponentin automaattinen testaus

Testi koostuu kolmesta osasta. Ensin testataan, että komponentti renderöi sille propsina annetun muuttujan tiedot oikein. Seuraavaksi testataan, että Poista- ja Muokkaa -napit kutsuvat tapahtumankäsittelijää vain kerran.

```
File: UserEdit.test.js  JS Note.test.js X  UserEdit.jsx U
c > Note > JS Note.test.js > ⚡ describe('<Note/>') callback > [⌚] mockHandler
1  import React from 'react'
2  import '@testing-library/jest-dom/extend-expect'
3  import { render, fireEvent } from '@testing-library/react'
4  import Note from './Note'
5
6  describe('<Note/>', () => {
7    let component
8
9    const note = {
10      noteId: "1111",
11      noteDate: '2022-04-01T09:42:00',
12      noteText: "Testing note component"
13    }
14
15    const mockHandler = jest.fn()
16
17    beforeEach(() => {
18      component = render(
19        <Note note={note} handleDeleteClick={mockHandler} handleEditClick={mockHandler} />
20      )
21    })
22
23    test('renders the date and text of note', () => {
24      const div = component.container.querySelector('.notepage')
25      expect(div).toHaveTextContent(
26        'Testing note component'
27      )
28      expect(component.container).toHaveTextContent(
29        'Päivämäärä: 1.4.2022 klo 09.42'
30      )
31    })
32
33    test('Button click only calls the event handler once', async () => {
34      const button = component.container.querySelector('#notedelete')
35      fireEvent.click(button)
36      expect(mockHandler.mock.calls).toHaveLength(1)
37    })
38
39    test('Button click only calls the event handler once', async () => {
40      const button = component.container.querySelector('#noteedit')
41      fireEvent.click(button)
42      expect(mockHandler.mock.calls).toHaveLength(1)
43    })
44  })

```

Odotettu tulos: Komponentti renderöi annetun muuttujan noteDate- ja noteText-tiedot sivulle oikein. Napit kutsuvat tapahtumankäsittelyjötä vain kerran.

3.2.1.1.2 NoteAdd-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelyää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS NoteAdd.test.js U X
src > Note > JS NoteAdd.test.js > ⚡ describe('<NoteAdd/>') callback
  1 import React from 'react'
  2 import { render, fireEvent } from '@testing-library/react'
  3 import '@testing-library/jest-dom/extend-expect'
  4 import NoteAdd from './NoteAdd'
  5
  6 describe('<NoteAdd/>', () => {
  7   let component
  8   const mockFunktio = jest.fn()
  9
 10  beforeEach(() => {
 11    component = render(
 12      <NoteAdd setAddNote={mockFunktio} setNotes={mockFunktio} notes={mockFunktio} setMessage={mockFunktio}
 13        setShowMessage={mockFunktio} setIsPositive={mockFunktio}/>
 14    )
 15  })
 16
 17  test('Eventhandler is called only once', async() => {
 18    const button = component.container.querySelector('#cancelnoteadd')
 19    fireEvent.click(button)
 20    expect(mockFunktio.mock.calls).toHaveLength(1)
 21  })
 22
 23  test('Form updates states and submit works', () => {
 24    const noteInput = component.container.querySelector('#notetext')
 25    const noteDate = component.container.querySelector('#notedate')
 26    const form = component.container.querySelector('form')
 27
 28    fireEvent.change(noteInput, {
 29      target: { value: 'Testnote' }
 30    });
 31
 32    fireEvent.change(noteDate, {
 33      target: {value: '2022-04-01T09:42:00'}
 34    })
 35
 36    fireEvent.submit(form)
 37
 38    setTimeout(() => {
 39      expect(mockFunktio.mock.calls).toHaveLength(1)
 40      expect(mockFunktio.mock.calls[0][0]).toBe('Testnote')
 41      expect(mockFunktio.mock.calls[0][1]).toHaveTextContent('Päivämäärä: 1.4.2022 klo 09.42')
 42    }, 0);
 43  })
 44})

```

Odottettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran.

Lomake päivittää submitilla annetut tiedot hooks-muuttujien stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.1.3 NoteEdit-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS NoteEdit.test.js U X
src > Note > JS NoteEdit.test.js > ⚡ describe("<NoteEdit/>") callback
  1 import React from 'react'
  2 import { render, fireEvent } from '@testing-library/react'
  3 import '@testing-library/jest-dom/extend-expect'
  4 import NoteEdit from './NoteEdit'
  5
  6 describe('<NoteEdit/>', () => {
  7   let component
  8   const mockFunktio = jest.fn()
  9
 10  beforeEach(() => {
 11    component = render(
 12      <NoteEdit setEditNote={mockFunktio} setNotes={mockFunktio} notes={mockFunktio} setMessage={mockFunktio}
 13        setShowMessage={mockFunktio} setIsPositive={mockFunktio} changedNote={mockFunktio}/>
 14    )
 15  })
 16
 17  test('Eventhandler is called only once', async() => {
 18    const button = component.container.querySelector('#cancelnoteedit')
 19    fireEvent.click(button)
 20    expect(mockFunktio.mock.calls).toHaveLength(1)
 21  })
 22
 23  test('Form updates states and submit works', () => {
 24    const noteInput = component.container.querySelector('#textedit')
 25    const noteDate = component.container.querySelector('#dateedit')
 26    const form = component.container.querySelector('form')
 27
 28    fireEvent.change(noteInput, {
 29      target: { value: 'Testnote' }
 30    });
 31
 32    fireEvent.change(noteDate, {
 33      target: { value: '2022-04-01T09:42:00' }
 34    })
 35
 36    fireEvent.submit(form)
 37
 38    setTimeout(() => {
 39      expect(mockFunktio.mock.calls).toHaveLength(1)
 40      expect(mockFunktio.mock.calls[0][0]).toBe('Testnote')
 41      expect(mockFunktio.mock.calls[0][1]).toHaveTextContent('Päivämäärä: 1.4.2022 klo 09.42')
 42    }, 0);
 43  })
 44})
```

Odotettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujien stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.2 *Location -kansion komponenttien automaattinen testaus*

3.2.1.2.1 Location-komponentin automaattinen testaus

Testi koostuu kolmesta osasta. Ensin testataan, että komponentti renderöi sille propsina annetun muuttujan tiedot oikein. Seuraavaksi testataan, että Poista- ja Muokkaa -napit kutsuvat tapahtumankäsittelijää vain kerran.

```
S PainlogEdit.test.js U JS Location.test.js X
src > Location > JS Location.test.js > ⚡ describe('<Location/>') callback > ⚡ test('Button click only calls the event handler once') callback > [x] button
  1 import React from 'react'
  2 import '@testing-library/jest-dom/extend-expect'
  3 import { render, fireEvent } from '@testing-library/react'
  4 import Location from './Location'
  5
  6 describe('<Location/>', () => {
  7   let component
  8
  9   const location = {
10     locationId: "1111",
11     locationName: "Testlocation"
12   }
13
14   const mockHandler = jest.fn()
15
16   beforeEach(() => {
17     component = render(
18       <Location location={location} handleDeleteClick={mockHandler} handleEditClick={mockHandler} />
19     )
20   })
21
22   test('renders the name and id of location', () => {
23
24     const div = component.container.querySelector('.notepage')
25     expect(div).toHaveTextContent(
26       'Testlocation'
27     )
28     expect(component.container).toHaveTextContent(
29       '1111'
30     )
31   })
32
33   test('Button click only calls the event handler once', async () => {
34     const button = component.container.querySelector('#locdelbutton')
35     fireEvent.click(button)
36     expect(mockHandler.mock.calls).toHaveLength(1)
37   })
38
39   test('Button click only calls the event handler once', async () => {
40     const button = component.container.querySelector('#loceditbutton')
41     fireEvent.click(button)
42     expect(mockHandler.mock.calls).toHaveLength(1)
43   })
44 })
```

Odotettu tulos: Komponentti renderöi annetun muuttujan locationId- ja locationName-tiedot sivulle oikein. Napit kutsuvat tapahtumankäsittelijötä vain kerran.

3.2.1.2.2 LocationAdd-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS LocationAdd.test.js U X
src > Location > JS LocationAdd.test.js > ⚡ describe('<LocationAdd/>') callback > [e] component
  1 import React from 'react'
  2 import { render, fireEvent } from '@testing-library/react'
  3 import '@testing-library/jest-dom/extend-expect'
  4 import LocationAdd from './LocationAdd'
  5
  6 describe('<LocationAdd/>', () => {
  7   let component
  8   const mockFunktio = jest.fn()
  9
 10  beforeEach(() => {
 11    component = render(
 12      <LocationAdd setAddLocation={mockFunktio} setLocations={mockFunktio} locations={mockFunktio} setMessage={mockFunktio}
 13        setShowMessage={mockFunktio} setIsPositive={mockFunktio}/>
 14    )
 15  })
 16
 17  test('Eventhandler is called only once', async() => {
 18    const button = component.container.querySelector('#cancellocadd')
 19    fireEvent.click(button)
 20    expect(mockFunktio.mock.calls).toHaveLength(1)
 21  })
 22
 23  test('Form updates states and submits the form', () => {
 24    const locationInput = component.container.querySelector('#locationInput')
 25    const form = component.container.querySelector('form')
 26
 27    fireEvent.change(locationInput, {
 28      target: { value: 'Testlocation' }
 29    });
 30    fireEvent.submit(form)
 31
 32    setTimeout(() => {
 33      expect(mockFunktio.mock.calls).toHaveLength(1) //ei toimi, tulee nollana...
 34      expect(mockFunktio.mock.calls[0][0]).toBe('Testlocation')
 35    }, 0);
 36  })
 37})
 38})
```

Odottettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran.

Lomake päivittää submitilla annetut tiedot hooks-muuttujan stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.2.3 LocationEdit-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS LocationEdit.test.js U X
src > Location > JS LocationEdit.test.js ...
1 import React from 'react'
2 import { render, fireEvent } from '@testing-library/react'
3 import '@testing-library/jest-dom/extend-expect'
4 import LocationEdit from './LocationEdit'
5
6 describe('<LocationEdit>', () => {
7   let component
8   const mockFunktio = jest.fn()
9
10  beforeEach(() => {
11    component = render(
12      <LocationEdit setEditLocation={mockFunktio} setLocations={mockFunktio} locations={mockFunktio} setMessage={mockFunktio}
13      setShowMessage={mockFunktio} setIsPositive={mockFunktio} changedLocation={mockFunktio} />
14    )
15  })
16
17  test('Eventhandler is called only once', async() => {
18    const button = component.container.querySelector('#cancelloccedit')
19    fireEvent.click(button)
20    expect(mockFunktio.mock.calls).toHaveLength(1)
21  })
22
23  test('Form updates states and submits the form', () => {
24    const locationInput = component.container.querySelector('input')
25    const form = component.container.querySelector('form')
26
27    fireEvent.change(locationInput, {
28      target: { value: 'Testlocation' }
29    });
30
31    fireEvent.submit(form)
32
33    setTimeout(() => {
34      expect(mockFunktio.mock.calls).toHaveLength(1) //ei toimi, tulee nollana...
35      expect(mockFunktio.mock.calls[0][0]).toBe('Testlocation')
36    }, 0);
37  })
38})
39)
```

Odotettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujien stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.3 PainLog -kansion komponenttien automaattinen testaus

3.2.1.3.1 PainLog-komponentin automaattinen testaus

Testi koostuu kolmesta osasta. Ensin testataan, että Poista- ja Muokkaa -napit kutsuvat tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentti renderöi sille propsina annetun muuttujan tiedot oikein.

```
JS Painlog.test.js U X
src > Painlog > JS Painlog.test.js > ⚡ describe('<Painlog/>') callback
  1 import React from 'react'
  2 import '@testing-library/jest-dom/extend-expect'
  3 import { render, fireEvent } from '@testing-library/react'
  4 import Painlog from './Painlog'
  5
  6 describe('<Painlog/>', () => {
  7   let component
  8   const log = {
  9     logId: '1111',
 10     painIntensity: '1',
 11     startTime: '2022-04-01T09:42:00',
 12     endTime: '2022-04-01T10:42:00',
 13     medication: "testMedication",
 14     locationInfo: "testLocation",
 15     painTrigger: "testTrigger",
 16     painType: "testType",
 17     locationId: '107',
 18     notes: "Testnotes"
 19   }
 20
 21   const mockHandler = jest.fn()
 22
 23   beforeEach(() => {
 24     component = render(
 25       <Painlog log={log} handleDeleteClick={mockHandler} handleEditClick={mockHandler} />
 26     )
 27   })
 28
 29   test('Button click only calls the event handler once', async () => {
 30     const button = component.container.querySelector('#logdelete')
 31     fireEvent.click(button)
 32     expect(mockHandler.mock.calls).toHaveLength(1)
 33   })
 34
 35   test('Button click only calls the event handler once', async () => {
 36     const button = component.container.querySelector('#logedit')
 37     fireEvent.click(button)
 38     expect(mockHandler.mock.calls).toHaveLength(1)
 39   })
 40
 41   test('renders content properly', () => {
 42     const div = component.container.querySelector('.notepage')
 43     expect(div).toHaveTextContent(
 44       '1.4.2022 klo 09.42'
 45     )
 46   })
 47 })
```

Odottettu tulos: Komponentti renderöi annetun muuttujan startTime-tiedon sivulle oikein. Napit kutsuvat tapahtumankäsittelijöitä vain kerran.

3.2.1.3.2 PainLogAdd-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS PainlogAdd.test.js U X
src > Painlog > JS PainlogAdd.test.js > ✘ describe('<PainlogAdd/>') callback > ✘ test('Painlog add form updates states and submits the form') callback
  1 import React from 'react'
  2 import { render, fireEvent } from '@testing-library/react'
  3 import '@testing-library/jest-dom/extend-expect'
  4 import PainlogAdd from './PainlogAdd'
  5
  6 describe('<PainlogAdd/>', () => {
  7   let component
  8   const mockFunktio = jest.fn()
  9
 10  beforeEach(() => {
 11    component = render(
 12      <PainlogAdd setAddPainlog={mockFunktio} setPainlogs={mockFunktio} painlogs={mockFunktio} setMessage={mockFunktio}
 13        setShowMessage={mockFunktio} setIsPositive={mockFunktio}/>
 14    )
 15  })
 16
 17  test('Cancel-button eventhandler is called only once', async() => {
 18    const button = component.container.querySelector('#cancellogadd')
 19    fireEvent.click(button)
 20    expect(mockFunktio.mock.calls).toHaveLength(1)
 21  })
 22
 23  test('Painlog add form updates states and submits the form', () => {
 24    const intensity = component.container.querySelector('#intensity')
 25    const startDate = component.container.querySelector('#starttime')
 26    const endDate = component.container.querySelector('#endtime')
 27    const form = component.container.querySelector('form')
 28
 29    fireEvent.change(intensity, {
 30      target: { value: '5' }
 31    });
 32    fireEvent.change(startDate, {
 33      target: { value: '2022-04-01T09:42:00' }
 34    });
 35    fireEvent.change(endDate, {
 36      target: { value: '2022-04-01T09:42:00' }
 37    });
 38    fireEvent.submit(form)
 39
 40    setTimeout(() => {
 41      expect(mockFunktio.mock.calls).toHaveLength(1)
 42      expect(mockFunktio.mock.calls[0][0]).toBe('5')
 43      expect(mockFunktio.mock.calls[0][1]).toHaveTextContent('1.4.2022 klo 09.42')
 44      expect(mockFunktio.mock.calls[0][1]).toHaveTextContent('1.4.2022 klo 09.42')
 45    }, 0);
 46  })
 47})
```

Odotettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujan stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.3.3 PainLogEdit-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS PainlogEdit.test.js ✘
src > Painlog > JS PainlogEdit.test.js > ⚡ describe('<PainlogEdit/>') callback > ⚡ test('Painlog edit form updates states and submits the form') callback
  1 import React from 'react'
  2 import { render, fireEvent } from '@testing-library/react'
  3 import '@testing-library/jest-dom/extend-expect'
  4 import PainlogEdit from './PainlogEdit'
  5
  6 describe('<PainlogEdit/>', () => {
  7   let component
  8   const mockFunktio = jest.fn()
  9
 10  beforeEach(() => {
 11    component = render(
 12      <PainlogEdit setEditPainlog={mockFunktio} setPainlogs={mockFunktio} painlogs={mockFunktio} setMessage={mockFunktio}>
 13        <div>{`<br><br>`}</div>
 14        <div>{`<br><br>`}</div>
 15      </PainlogEdit>
 16    )
 17    test('Cancel-button eventhandler is called only once', async() => {
 18      const button = component.container.querySelector('#canceleditlog')
 19      fireEvent.click(button)
 20      expect(mockFunktio.mock.calls).toHaveLength(1)
 21    })
 22
 23    test('Painlog edit form updates states and submits the form', () => {
 24      const intensity = component.container.querySelector('#painintensity')
 25      const startDate = component.container.querySelector('#startdate')
 26      const endDate = component.container.querySelector('#enddate')
 27      const form = component.container.querySelector('form')
 28
 29      fireEvent.change(intensity, {
 30        target: { value: '5' }
 31      });
 32      fireEvent.change(startDate, {
 33        target: { value: '2022-04-01T09:42:00' }
 34      })
 35      fireEvent.change(endDate, {
 36        target: { value: '2022-04-01T09:42:00' }
 37      })
 38      fireEvent.submit(form)
 39
 40      setTimeout(() => {
 41        expect(mockFunktio.mock.calls).toHaveLength(1)
 42        expect(mockFunktio.mock.calls[0][0]).toBe('5')
 43        expect(mockFunktio.mock.calls[0][1]).toHaveTextContent('1.4.2022 klo 09.42')
 44        expect(mockFunktio.mock.calls[0][2]).toHaveTextContent('1.4.2022 klo 09.42')
 45      }, 0);
 46    })
 47  })
})
```

Odottettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujien stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.4 User -kansion komponenttien automaattiset testit

3.2.1.4.1 User-komponentin automaattinen testaus

Testi koostuu kolmesta osasta. Ensin testataan, että Poista- ja Muokkaa -napit kutsuvat tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentti renderöi sille propsina annetun muuttujan tiedot oikein.

```
JS User.test.js U X
src > Users > JS User.test.js > ⚡ describe('<User/>') callback
  1 import React from 'react'
  2 import '@testing-library/jest-dom/extend-expect'
  3 import { render, fireEvent } from '@testing-library/react'
  4 import User from './User'
  5
  6 describe('<User/>', () => {
  7   let component
  8
  9   const user = {
10     username: 'JestTest'
11   }
12
13   const mockHandler = jest.fn()
14
15   beforeEach(() => {
16     component = render(
17       <User user={user} handleDeleteClick={mockHandler} handleEditClick={mockHandler} />
18     )
19   })
20
21   test('Button click only calls the event handler once', async () => {
22     const button = component.container.querySelector('#userdelete')
23     fireEvent.click(button)
24     expect(mockHandler.mock.calls).toHaveLength(1)
25   })
26
27   test('Button click only calls the event handler once', async () => {
28     const button = component.container.querySelector('#useredit')
29     fireEvent.click(button)
30     expect(mockHandler.mock.calls).toHaveLength(1)
31   })
32
33   test('renders the page content', () => {
34
35     const div = component.container.querySelector('.notepage')
36     expect(div).toHaveTextContent(
37       'JestTest'
38     )
39   })
40 })
```

Odottettu tulos: Komponentti renderöi annetun muuttujan username-tiedon sivulle oikein. Napit kutsuvat tapahtumankäsittelijöitä vain kerran.

3.2.1.4.2 UserAdd-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS UserAdd.test.js U
src > Users > JS UserAdd.test.js > ⚡ describe('<UserAdd/>') callback > ⚡ test('Form updates states and submits the form') callback > ✎ userinput
  6  describe('<UserAdd/>', () => {
  7    let component
  8    const mockFunktio = jest.fn()
  9
 10   beforeEach(() => {
 11     component = render(
 12       <UserAdd setAddUser={mockFunktio} setUsers={mockFunktio} users={mockFunktio} setMessage={mockFunktio}
 13         setShowMessage={mockFunktio} setIsPositive={mockFunktio}>
 14     )
 15   })
 16
 17   test('Eventhandler is called only once', async() => {
 18     const button = component.container.querySelector('#canceluseradd')
 19     fireEvent.click(button)
 20     expect(mockFunktio.mock.calls).toHaveLength(1)
 21   })
 22
 23   test('Form updates states and submits the form', () => {
 24     const userInput = component.container.querySelector('#userinput')
 25     const form = component.container.querySelector('form')
 26
 27     fireEvent.change(userInput, {
 28       target: { value: 'JestTest' }
 29     });
 30
 31     fireEvent.submit(form)
 32
 33     setTimeout(() => {
 34       expect(mockFunktio.mock.calls).toHaveLength(1)
 35       expect(mockFunktio.mock.calls[0][0]).toBe('JestTest')
 36     }, 0);
 37   })
 38 })
```

Odotettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujan stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.1.4.3 UserEdit-komponentin automaattinen testaus

Testi koostuu kahdesta osasta. Ensin testataan, että Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Seuraavaksi testataan, että komponentin lomake päivittää hooks-muuttujien statet ja lomakkeen submit toimii.

```
JS UserEdit.test.js ×
src > Users > JS UserEdit.test.js > ⚡ describe('<UserEdit/>') callback > ⚡ test('Form updates states and submits the form') callback > [o] emailInput
17   test('Eventhandler is called only once', async() => {
18     const button = component.container.querySelector('#canceluseredit')
19     fireEvent.click(button)
20     expect(mockFunktio.mock.calls).toHaveLength(1)
21   })
22
23   test('Form updates states and submits the form', () => {
24     const emailInput = component.container.querySelector('#emailinput')
25     const passwordInput = component.container.querySelector('#passwordinput')
26     const passwordAgainInput = component.container.querySelector('#passwordagaininput')
27     const form = component.container.querySelector('form')
28
29     fireEvent.change(emailInput, {
30       target: { value: 'Jest@Test' }
31     });
32
33     fireEvent.change(passwordInput, {
34       target: { value: 'Jest' }
35     });
36
37     fireEvent.change(passwordAgainInput, {
38       target: { value: 'Jest' }
39     });
40
41     fireEvent.submit(form)
42
43     setTimeout(() => {
44       expect(mockFunktio.mock.calls).toHaveLength(1)
45       expect(mockFunktio.mock.calls[0][0]).toBe('Jest@Test')
46     }, 0);
47   })
48 })
```

Odottettu tulos: Komponentin Peruuta-nappi kutsuu tapahtumankäsittelijää vain kerran. Lomake päivittää submitilla annetut tiedot hooks-muuttujien stateen ja submit kutsuu tapahtumankäsittelijää.

3.2.2 Integraatiotestaus Alba -kirjastolla

Kipukalenteri-sovelluksen integraatiotestaus suoritetaan .NET Core Rest API -sovelluksessa Alba- ja xunit-kirjastoilla testaamalla back end -projektiin ja tietokannan välistä yhteyttä. Testauksen kohteena ovat Note-, PainLocation-, PainLog- ja UserControllerien metodit HttpGet, HttpPut, HttpPost ja HttpDelete.

3.2.2.1 NoteControllerin integraatiotestaus

Testitiedosto koostuu viidestä yksittäisestä testistä. Kaikki testit on suoritettu samalla kertaa onnistuneesti, testien tulokset vastasivat alla testitapauskuvauksissa asetettuja odotettuja tuloksia.

3.2.2.1.1 GetAll()-metodin testaus

Testataan, että kontrollerin GetAll()-metodi toimii ja tuloksena saatu tieto ei ole tyhjä

```

9   namespace TestKipukalenteriBackend
10  {
11      public class UnitTestNote
12      {
13          [Fact]
14          public async Task GetAll()
15          {
16              var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
17              using var system = new AlbaHost(hostBuilder);
18              {
19                  await system.Scenario(s =>
20                  {
21                      s.Get.Url("/kipukalenteri/note");
22                      s.StatusCodeShouldBeOk();
23                  });
24
25                  var results = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/note");
26                  Assert.NotEmpty(results);
27              }
28          }
}

```

Odotettu tulos: Tietojen haku GetAll()-metodilla toimii, tuloksena saatu tieto ei ole tyhjä.

3.2.2.1.2 GetOne() -metodin testaus

Testataan, että GetOne()-metodi toimii. Hakua testataan kolmella tavalla:

Haetaan Id:llä, jota tietokantataulussa ei ole

Haetaan Id:llä, joka taulusta löytyy

Haetaan Id:llä, joka on väärän tyyppinen

```

[Fact]
public async Task GetOne()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/note/7554");
            s.StatusCodeShouldBe(204);

            s.Get.Url("/kipukalenteri/note/101");
            s.StatusCodeShouldBe(200);

            s.Get.Url("/kipukalenteri/note/note");
            s.StatusCodeShouldBe(400);
        });
    }
}

```

Odotettu tulos: Ensimmäisen kohdan pyynnöstä saadaan vastaukseksi status-koodi 204 No Content, toisen kohdan pyynnöstä saadaan status-koodi 200 OK ja kolmannen kohdan pyynnöstä status-koodi 400 Bad Request.

3.2.2.1.3 AddNewWithoutBody() - Create()-metodin testaus

Lähetetään HttpPost -metodille lisäyspyyntö laittamatta Body-osioon lähetettävän muistiinpanon tietoja.

```
52  | 0 references
53  public async Task AddNewWithoutBody()
54  {
55      var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
56      using var system = new AlbaHost(hostBuilder);
57      {
58          await system.Scenario(s =>
59          {
60              s.Post.Url("/kipukalenteri/note");
61              s.StatusCodeShouldBe(415);
62          });
63      }
}
```

Odotettu tulos: Vastauksena saadaan status-koodi 415 Unsupported Media Type, lisäys ei onnistuu.

3.2.2.1.4 AddNew() - Create()-metodin testaus

Lähetetään HttpPost-metodilla lisäyspyyntö ja Body-osiossa lähetettävät tiedot.

```
65  | [Fact]
66  | 0 references
67  public async Task AddNew()
68  {
69      var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
70      using var system = new AlbaHost(hostBuilder);
71      {
72          var results1 = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/note");
73          var count1 = results1.Count();
74
75          var note = new Note
76          {
77              NoteDate = DateTime.Parse("2022-04-01T09:42:00"), NoteText = "Testing note controller", UserId = "testiad"
78          };
79
80          await system.Scenario(s =>
81          {
82              s.Post.Json<Note>(note).ToUrl("/kipukalenteri/note");
83              s.StatusCodeShouldBe(200);
84          });
85
86      }
}
```

Odotettu tulos: Lisäys onnistuu ja vastauksena saadaan status-koodi 200 OK. Notes-taulun tulosten määrä on lisääntynyt yhdellä ennen lisäystä lasketusta määrästä.

3.2.2.1.5 Delete() -metodin testaus

Poistetaan viimeisimmäksi lisätty rivi tietokantataulusta HttpDelete-metodilla.

```

88     [Fact]
89     public async Task DeleteLatest()
90     {
91         var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
92         using var system = new AlbaHost(hostBuilder);
93         {
94
95             var results1 = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/note");
96             var count1 = results1.Count();
97             int biggestId = (from r in results1 select r.NoteId).ToArray().Max();
98
99             await system.Scenario(s =>
100             {
101                 s.Delete.Url("/kipukalenteri/note/" + biggestId);
102                 s.StatusCodeShouldBe(200);
103             });
104
105             var results2 = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/note");
106             Assert.Equal(count1 - 1, results2.Count());
107         }
108     }
109

```

Odotettu tulos: Poisto onnistuu, vastauksena saadaan status-koodi 200 OK ja taulun tulosten määrä on vähenytynyt yhdellä ennen poistoa lasketusta määristää.

3.2.2.2 PainLocationControllerin integraatiotestaus

Testitiedosto koostuu viidestä yksittäisestä testistä. Kaikki testit on suoritettu samalla kertaa onnistuneesti, testien tulokset vastasivat alla testitapauskuvauksissa asetettuja odotettuja tuloksia.

3.2.2.2.1 GetAll()-metodin testaus

Testataan, että kontrollerin GetAll()-metodi toimii ja tuloksena saatu tieto ei ole tyhjä

```

0 references
public class UnitTestPainlocation
{
    [Fact]
    public async Task GetAll()
    {
        var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
        using var system = new AlbaHost(hostBuilder);
        {
            await system.Scenario(s =>
            {
                s.Get.Url("/kipukalenteri/painlocation");
                s.StatusCodeShouldBeOk();
            });

            var results = await system.GetAsJson<IEnumerable<PainLocation>>("/kipukalenteri/painlocation");
            Assert.NotEmpty(results);
        }
    }
}

```

Odotettu tulos: Tietojen haku GetAll()-metodilla toimii, tuloksena saatu tieto ei ole tyhjä.

3.2.2.2.2 GetOne() -metodin testaus

Testataan, että GetOne()-metodi toimii. Hakua testataan kolmella tavalla:

Haetaan Id:llä, jota tietokantataulussa ei ole

Haetaan Id:llä, joka taulusta löytyy

Haetaan Id:llä, joka on väärän tyyppinen

```
[Fact]
0 | 0 references
public async Task GetOne()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/painlocation/7554");
            s.StatusCodeShouldBe(204);

            s.Get.Url("/kipukalenteri/painlocation/105");
            s.StatusCodeShouldBe(200);

            s.Get.Url("/kipukalenteri/painlocation/location");
            s.StatusCodeShouldBe(400);
        });
    }
}
```

Odotettu tulos: Ensimmäisen kohdan pyynnöstä saadaan vastaukseksi status-koodi 204 No Content, toisen kohdan pyynnöstä saadaan status-koodi 200 OK ja kolmannen kohdan pyynnöstä status-koodi 400 Bad Request.

3.2.2.2.3 AddNewWithoutBody() - Create()-metodin testaus

Lähetetään HttpPost -metodille lisäyspyyntö laittamatta Body-osioon lähetettävän sijainnin tietoja.

```
[Fact]
0 | 0 references
public async Task AddNewWithoutBody()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Post.Url("/kipukalenteri/painlocation");
            s.StatusCodeShouldBe(415);
        });
    }
}
```

Odotettu tulos: Vastauksena saadaan status-koodi 415 Unsupported Media Type, lisäys ei onnistu.

3.2.2.2.4 AddNew() - Create()-metodin testaus

Lähetetään HttpPost-metodilla lisäyspyyntö ja Body-osiossa lähetettävät tiedot.

```

65     [Fact]
66     public async Task AddNew()
67     {
68         var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
69         using var system = new AlbaHost(hostBuilder);
70         {
71             var results1 = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/painlocation");
72             var count1 = results1.Count();
73
74             var location = new PainLocation
75             { LocationName = "TestingAlba" };
76
77             await system.Scenario(s =>
78             {
79                 s.Post.Json<PainLocation>(location).ToUrl("/kipukalenteri/painlocation");
80                 s.StatusCodeShouldBe(200);
81             });
82
83             var results2 = await system.GetAsJson<IEnumerable<Note>>("/kipukalenteri/painlocation");
84             Assert.Equal(count1 + 1, results2.Count());
85         }
86     }
87 
```

Odotettu tulos: Lisäys onnistuu ja vastauksena saadaan status-koodi 200 OK. PainLocation-taulun tulosten määrä on lisääntynyt yhdellä ennen lisäystä lasketusta määrästä.

3.2.2.2.5 Delete() -metodin testaus

Poistetaan viimeisimmäksi lisätty rivi tietokantataulusta HttpDelete-metodilla.

```

88     [Fact]
89     public async Task DeleteLatest()
90     {
91         var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
92         using var system = new AlbaHost(hostBuilder);
93         {
94
95             var results1 = await system.GetAsJson<IEnumerable<PainLocation>>("/kipukalenteri/painlocation");
96             var count1 = results1.Count();
97
98             int biggestId = (from r in results1 select r.LocationId).ToArray().Max();
99
100            await system.Scenario(s =>
101            {
102                s.Delete.Url("/kipukalenteri/painlocation/" + biggestId);
103                s.StatusCodeShouldBe(200);
104            });
105
106            var results2 = await system.GetAsJson<IEnumerable<PainLocation>>("/kipukalenteri/painlocation");
107            Assert.Equal(count1 - 1, results2.Count());
108        }
109    }
110 }
111 
```

Odotettu tulos: Poisto onnistuu, vastauksena saadaan status-koodi 200 OK ja taulun tulosten määrä on vähentynyt yhdellä ennen poistoa lasketusta määrästä.

3.2.2.3 PainLogControllerin integraatiotestaus

Testitiedosto koostuu viidestä yksittäisestä testistä. Kaikki testit on suoritettu samalla kertaa onnistuneesti, testien tulokset vastasivat alla testitapauskuvauksissa asetettuja odotettuja tuloksia.

3.2.2.3.1 GetAll()-metodin testaus

Testataan, että kontrollerin GetAll()-metodi toimii ja tuloksena saatu tieto ei ole tyhjä

```
public async Task GetAll()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/painlog");
            s.StatusCodeShouldBeOk();
        });

        var results = await system.GetAsJson("/kipukalenteri/painlog");
        Assert.NotEmpty(results);
    }
}
```

Odotettu tulos: Tietojen haku GetAll()-metodilla toimii, tuloksesta saatu tieto ei ole tyhjä.

3.2.2.3.2 GetOne() -metodin testaus

Testataan, että GetOne()-metodi toimii. Hakua testataan kolmella tavalla:

Haetaan Id:llä, jota tietokantataulussa ei ole

Haetaan Id:llä, joka taulusta löytyy

Haetaan Id:llä, joka on väärän tyypinen

```
[Fact]
➊ | 0 references
public async Task GetOne()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/painlog/75544");
            s.StatusCodeShouldBe(204);

            s.Get.Url("/kipukalenteri/painlog/10114");
            s.StatusCodeShouldBe(200);

            s.Get.Url("/kipukalenteri/painlog/painlog");
            s.StatusCodeShouldBe(400);
        });
    }
}
```

Odotettu tulos: Ensimmäisen kohdan pyynnöstä saadaan vastaukseksi status-koodi 204 No Content, toisen kohdan pyynnöstä saadaan status-koodi 200 OK ja kolmannen kohdan pyynnöstä status-koodi 400 Bad Request.

3.2.2.3.3 AddNewWithoutBody() - Create()-metodin testaus

Lähetetään HttpPost -metodille lisäyspyyntö laittamatta Body-osioon lähetettävän sijainnin tietoja.

```

● | 0 references
public async Task AddNewWithoutBody()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Post.Url("/kipukalenteri/painlog");
            s.StatusCodeShouldBe(415);
        });
    }
}

```

Odottettu tulos: Vastauksena saadaan status-koodi 415 Unsupported Media Type, lisäys ei onnistuu.

3.2.2.3.4 AddNew() - Create()-metodin testaus

Lähetetään HttpPost-metodilla lisäyspyyntö ja Body-osiossa lähetettävät tiedot.

```

[Fact]
● | 0 references
public async Task AddNew()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {

        var results1 = await system.GetAsJson<IEnumerable<PainLog>>("/kipukalenteri/painlog");
        var count1 = results1.Count();

        var painlog = new PainLog
        {
            PainIntensity = 5,
            StartTime = DateTime.Parse("2022-04-01T09:42:00"),
            EndTime = DateTime.Parse("2022-04-01T09:42:00"),
            Medication = "",
            LocationInfo = "",
            PainTrigger = "",
            PainType = "",
            LocationId = 105,
            Notes = "",
            UserId = "testiad"
        };

        await system.Scenario(s =>
        {
            s.Post.Json<PainLog>(painlog).ToUrl("/kipukalenteri/painlog");
            s.StatusCodeShouldBe(200);
        });

        var results2 = await system.GetAsJson<IEnumerable<PainLog>>("/kipukalenteri/painlog");
        Assert.Equal(count1 + 1, results2.Count());
    }
}

```

Odottettu tulos: Lisäys onnistuu ja vastauksena saadaan status-koodi 200 OK. PainLog-taulun tulosten määrä on lisääntynyt yhdellä ennen lisäystä lasketusta määrästä.

Tulos: Testi toteutui odotetun tuloksen mukaisesti.

3.2.2.3.5 Delete() -metodin testaus

Poistetaan viimeisimmäksi lisätty rivi tietokantataulusta HttpDelete-metodilla.

```

100
101 [Fact]
102 public async Task DeleteLatest()
103 {
104     var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
105     using var system = new AlbaHost(hostBuilder);
106
107     var results1 = await system.GetAsJson<IEnumerable<PainLog>>("/kipukalenteri/painlog");
108     var count1 = results1.Count();
109
110     int biggestId = (from r in results1 select r.LogId).ToArray().Max();
111
112     await system.Scenario(s =>
113     {
114         s.Delete.Url("/kipukalenteri/painlog/" + biggestId);
115         s.StatusCodeShouldBe(200);
116     });
117
118     var results2 = await system.GetAsJson<IEnumerable<PainLog>>("/kipukalenteri/painlog");
119     Assert.Equal(count1 - 1, results2.Count());
120 }
121
122
123
124
125 }
```

Odottettu tulos: Poisto onnistuu, vastauksena saadaan status-koodi 200 OK ja taulun tulosten määrä on vähentynyt yhdellä ennen poistoa lasketusta määristää.

3.2.2.4 UserControllerin integraatiotestaus

Testitiedosto koostuu viidestä yksittäisestä testistä. Kaikki testit on suoritettu samalla kertaa onnistuneesti, testien tulokset vastasivat alla testitapauskuvauksissa asetettuja odotettuja tuloksia.

3.2.2.4.1 GetAll()-metodin testaus

Testataan, että kontrollerin GetAll()-metodi toimii ja tuloksena saatu tieto ei ole tyhjä

```

[Fact]
public async Task GetAll()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/user");
            s.StatusCodeShouldBeOk();
        });

        var results = await system.GetAsJson<IEnumerable<User>>("/kipukalenteri/user");
        Assert.NotEmpty(results);
    }
}
```

Odottettu tulos: Tietojen haku GetAll()-metodilla toimii, tuloksena saatu tieto ei ole tyhjä.

3.2.2.4.2 GetOne() -metodin testaus

Testataan, että GetOne()-metodi toimii. Hakua testataan kahdella tavalla:

Haetaan Id:llä, jota tietokantataulussa ei ole

Haetaan Id:llä, joka taulusta löytyy

```
[Fact]
➊ | 0 references
public async Task GetOne()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Get.Url("/kipukalenteri/user/7554");
            s.StatusCodeShouldBe(204);

            s.Get.Url("/kipukalenteri/note/anna1");
            s.StatusCodeShouldBe(200);
        });
    }
}
```

Odotettu tulos: Ensimmäisen kohdan pyynnöstä saadaan vastaukseksi status-koodi 204 No Content ja toisen kohdan pyynnöstä saadaan status-koodi 200 OK.

3.2.2.4.3 AddNewWithoutBody() - Create()-metodin testaus

Lähetetään HttpPost -metodille lisäyspyyntö laittamatta Body-osioon lähetettävän sijainnin tietoja.

```
[Fact]
➊ | 0 references
public async Task AddNewWithoutBody()
{
    var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
    using var system = new AlbaHost(hostBuilder);
    {
        await system.Scenario(s =>
        {
            s.Post.Url("/kipukalenteri/user");
            s.StatusCodeShouldBe(415);
        });
    }
}
```

Odotettu tulos: Vastauksena saadaan status-koodi 415 Unsupported Media Type, lisäys ei onnistu.

3.2.2.4.4 AddNew() - Create()-metodin testaus

Lähetetään HttpPost-metodilla lisäyspyyntö ja Body-osiossa lähetettävät tiedot.

```

61 [Fact]
62     public async Task AddNew()
63     {
64         var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
65         using var system = new AlbaHost(hostBuilder);
66         {
67             var results1 = await system.GetAsJson<IEnumerable<User>>("/kipukalenteri/user");
68             var count1 = results1.Count();
69
70             var user = new User
71             { Username = "AlbaUser", Email = "Alba@testi", Password = "Alba1" };
72
73             await system.Scenario(s =>
74             {
75                 s.Post.Json<User>(user).ToUrl("/kipukalenteri/user");
76                 s.StatusCodeShouldBe(200);
77             });
78
79             var results2 = await system.GetAsJson<IEnumerable<User>>("/kipukalenteri/user");
80             Assert.Equal(count1 + 1, results2.Count());
81         }
82     }
83

```

Odotettu tulos: Lisäys onnistuu ja vastauksena saadaan status-koodi 200 OK. Users-taulun tulosten määrä on lisääntynyt yhdellä ennen lisäystä lasketusta määrästä.

Tulos: Testi toteutui odotetun tuloksen mukaisesti.

3.2.2.4.5 Delete() -metodin testaus

Poistetaan viimeisimmäksi lisätty rivi tietokantataulusta HttpDelete-metodilla.

```

84 [Fact]
85     public async Task DeleteLatest()
86     {
87         var hostBuilder = Kipukalenteri.Program.CreateHostBuilder(new string[0]);
88
89         using var system = new AlbaHost(hostBuilder);
90         {
91
92             var results1 = await system.GetAsJson<IEnumerable<User>>("/kipukalenteri/user");
93             var count1 = results1.Count();
94
95             await system.Scenario(s =>
96             {
97                 s.Delete.Url("/kipukalenteri/user/" + "AlbaUser");
98                 s.StatusCodeShouldBe(200);
99             });
100
101            var results2 = await system.GetAsJson<IEnumerable<User>>("/kipukalenteri/user");
102            Assert.Equal(count1 - 1, results2.Count());
103        }
104    }
105
106
107

```

Odotettu tulos: Poisto onnistuu, vastauksena saadaan status-koodi 200 OK ja taulun tulosten määrä on vähentynyt yhdellä ennen poistoa lasketusta määrästä.

3.2.3 End-to-end testaus Cypress-kirjastolla

Kipukalenteri-sovelluksen end-to-end -testaus suoritetaan react.js-projektissa Cypress - testikirjastolla. Testauksen kohteena ovat Etusivun sekä Sijainnit-, Muistiinpanot- ja

Kipumerkinnät-näkymien toiminnot. Testauksessa testataan tietojen lisäämistä ja poistamista sovelluksesta. Testauksen tulos on käsitelty testauspöytäkirjassa.

3.2.3.1 Muistiinpanot-näkymän testaus

Testitiedosto koostuu kolmesta osasta.

Ennen jokaista testiä suoritetaan beforeEach -osuus eli siirrytään sivulle

<http://localhost:3000/Note>

```
cypress > integration > js note.spec.js > ...
1  describe('Note', function () {
2    beforeEach(function () {
3      cy.visit('http://localhost:3000/Note')
4    })
5  })
```

3.2.3.1.1 Sivu avautuu ja renderöi tietokantataulun sisällön

Testataan, että Muistiinpanot-sivu renderöi "Muistiinpanot" -otsikon lisäksi tietokannasta dataa sivulle.

```
6  it('Site opens', function() {
7    cy.contains('Muistiinpanot')
8    cy.contains('Lisää')
9    cy.contains('Päivämäärä: 24.4.2022')
10   cy.contains('Testimuistiinpano')
11 })
```

Odottettu tulos: Sivu avautuu ja renderöi määritellyt tiedot.

3.2.3.1.2 Muistiinpanon lisäys toimii

Testataan, että muistiinpanon lisäys toimii, lisätty muistiinpano siirtyy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

```
13  it('Add Note works right', function () {
14    cy.contains('Lisää').click()
15    cy.contains('Tallenna')
16    cy.contains('Peruuta')
17    cy.get('#notedate').type('2022-05-14T09:42')
18    cy.get('#notetext').type('Cypress-testNote')
19    cy.get('#submitnote').click()
20    cy.contains('Cypress-testNote')
21    cy.get('.notetext').first().contains('Cypress-testNote')
22  })
23
```

Odottettu tulos: Muistiinpanon lisäys toimii, lisätty muistiinpano siirtyy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

3.2.3.1.3 Muistiinpanon poisto toimii

Testataan, että muistiinpanon poisto toimii, eikä poistettu muistiinpano enää renderöidy sivulle.

```
24     it('Remove Note works', function () {
25       cy.get('.buttons').first().contains('Poista').click()
26       cy.get('.notetext').first().should('not.contain', 'Cypress-testNote')
27     })
28   })
29 })
```

Odotettu tulos: Muistiinpanon poisto toimii, eikä poistettu muistiinpano enää renderöidy sivulle.

3.2.3.2 Sijainnit-näkymän testaus

Testitiedosto koostuu kolmesta osasta.

Ennen jokaista testiä suoritetaan beforeEach -osuus eli kirjaudutaan sisään ja siirrytään sivulle <http://localhost:3000/Location>

```
6    beforeEach(function () {
7      cy.visit('http://localhost:3000/')
8      cy.get('#hpuserinput').type('testiad')
9      cy.get('#hpuserpass').type('Kaapeli01')
10     cy.get('#submitlogin').click()
11     cy.wait(500)
12     cy.visit('http://localhost:3000/Location')
13   })
```

3.2.3.2.1 Sivu avautuu ja renderöi tietokantataulun sisällön

Testataan, että Sijainnit-sivu renderöi "Kivun sijainti" -otsikon lisäksi tietokannasta dataa sivulle.

```
15   it('Site opens', function() {
16     cy.contains('Kivun sijainti')
17     cy.contains('ID 104')
18     cy.contains('Pää')
19   })
```

Odotettu tulos: Sivu avautuu ja renderöi määritellyt tiedot.

3.2.3.2.2 Sijainnin lisäys toimii

Testataan, että sijainnin lisäys toimii, lisätty sijainti siirtyy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

```
21 |     it('Add Location works right', function () {
22 |       cy.wait(500)
23 |       cy.get('.nappi').click()
24 |       cy.wait(500)
25 |       cy.contains('Lisää sijainti')
26 |       cy.get('#locationInput').type('Cypress-testLocation')
27 |       cy.get('#submitlocation').click()
28 |       cy.wait(500)
29 |       cy.get('.notepage').last().contains('Cypress-testLocation')
30 |     })
31 |   })
```

Odotettu tulos: Sijainnin lisäys toimii, lisätty sijainti siirtyy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

3.2.3.2.3 Sijainnin poisto toimii

Testataan, että sijainnin poisto toimii, eikä poistettu sijainti enää renderöidy sivulle.

```
32 |     it('Remove Location works', function () {
33 |       cy.get('#locdelbutton').last().click()
34 |       cy.wait(500)
35 |       cy.get('div').last().should('not.contain', 'Cypress-testLocation')
36 |     })
37 |   })
```

Odotettu tulos: Sijainnin poisto toimii, eikä poistettu sijainti enää renderöidy sivulle.

3.2.3.3 Kipumerkinnät-näkymän testaus

Testitiedosto koostuu kolmesta osasta.

Ennen jokaista testiä suoritetaan beforeEach -osuus eli siirrytään sivulle

<http://localhost:3000/Painlog>

```
cypress > integration > JS painlog.spec.js > ...
1  describe('Painlog', function () {
2    beforeEach(function () {
3      cy.visit('http://localhost:3000/Painlog')
4    })
5  })
```

3.2.3.3.1 Sivu avautuu ja renderöi tietokantataulun sisällön

Testataan, että Kipumerkinnät-sivu renderöi "Kipumerkinnät" -otsikon lisäksi tietokannasta dataa sivulle.

```
6  it('Site opens', function() {
7    cy.contains('Kipumerkinnät')
8    cy.contains('Lisää')
9    cy.contains('Alkupäivämäärä: 11.5.2022')
10   cy.contains('Kivun sijainti: Pää')
11 })
12 })
```

Odottettu tulos: Sivu avautuu ja renderöi määritellyt tiedot.

3.2.3.3.2 Kipumerkinnän lisäys toimii

Testataan, että kipumerkinnän lisäys toimii, lisätty kipumerkintä siirryy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

```
13     it('Add Painlog works right', function () {
14       cy.contains('Lisää').click()
15       cy.contains('Tallenna')
16       cy.contains('Peruuta')
17       cy.get('#intensity').type(5)
18       cy.get('#starttime').type('2022-05-14T09:42')
19       cy.get('#endtime').type('2022-05-14T10:42')
20       cy.get('#locationselect').select('107')
21       cy.get('#submitpainlog').click()
22       cy.get('.notepage').first().contains('Alkupäivämäärä: 14.5.2022 ')
23     })
```

Odottettu tulos: Merkinnän lisäys toimii, lisätty merkintä siirryy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

3.2.3.3.3 Kipumerkinnän poisto toimii

Testataan, että kipumerkinnän poisto toimii, eikä poistettu merkintä enää renderöidy sivulle.

```
25     it('Remove Painlog works', function () {
26       cy.get('.buttons').first().contains('Poista').click()
27       cy.get('.notepage').first().should('not.contain', 'Käsi')
28     })
29
30   })
```

Odottettu tulos: Merkinnän poisto toimii, eikä poistettu merkintä enää renderöidy sivulle.

3.2.3.4 Frontpage-näkymän testaus

Testitiedosto koostuu kolmesta osasta.

Ennen jokaista testiä suoritetaan beforeEach -osuus eli kirjaudutaan sisään

```
6     beforeEach(function () {
7       cy.visit('http://localhost:3000/')
8       cy.get('#hpuserinput').type('testiad')
9       cy.get('#hpuserpass').type('Kaapeli01')
10      cy.get('#submitlogin').click()
11      cy.wait(500)
12    })
```

3.2.3.4.1 Kirjautuminen toimii ja sivu avautuu ja renderöi sisällön

Testataan, että kirjautuminen on onnistunut ja etusivu renderöityy oikein.

```
14     it('Login works and site opens', function() {
15         cy.contains('Kirjautuneena')
16         cy.contains('Lisää kipumerkintä')
17         cy.contains('Lisää sijainti')
18     })
```

Odotettu tulos: Sivulta löytyy teksti "Kirjautuneena" eli kirjautuminen on onnistunut ja sivulta löytyy napit "Lisää kipumerkintä" ja "Lisää sijainti".

3.2.3.4.2 Sijainnin lisäys toimii

Testataan, että sijainnin lisäys toimii, lisätty sijainti siirtyy tietokantaan ja lisättyt tiedot renderöityvät Sijainnit-sivulle.

```
20     it('Add Location works right', function () {
21         cy.get('#hpaddloc').click()
22         cy.contains('Lisää sijainti')
23         cy.get('#locationInput').type('Cypress-testFrontpage')
24         cy.get('#submitlocation').click()
25         cy.wait(500)
26         cy.visit('http://localhost:3000/Location')
27         cy.contains('Postman API test 2')
28         cy.get('.notepage').last().contains('Cypress-testFrontpage')
29     })
```

Odotettu tulos: Sijainnin lisäys toimii, lisätty sijainti siirtyy tietokantaan ja lisättyt tiedot renderöityvät sivulle.

3.2.3.4.3 Merkinnän lisäys toimii

Testataan, että kipumerkinnän lisäys toimii, lisätty merkintä siirtyy tietokantaan ja lisättyt tiedot renderöityvät Kipumerkinnät-sivulle.

```
31     it('Add Painlog works right', function () {
32         cy.get('#hpaddpainlog').click()
33         cy.contains('Lisää kipumerkintä')
34         cy.get('#intensity').type(5)
35         cy.get('#starttime').type('2022-06-05T09:42')
36         cy.get('#endtime').type('2022-06-05T10:42')
37         cy.get('#locationselect').select('105')
38         cy.get('#submitpainlog').click()
39         cy.wait(500)
40         cy.visit('http://localhost:3000/Painlog')
41         cy.get('.notepage').first().contains('Alkupäivämäärä: 5.6.2022 ')
42         cy.get('.notepage').first().contains('Pää')
43     })
```

Odotettu tulos: Sijainnin poisto toimii, eikä poistettu sijainti enää renderöidy sivulle.

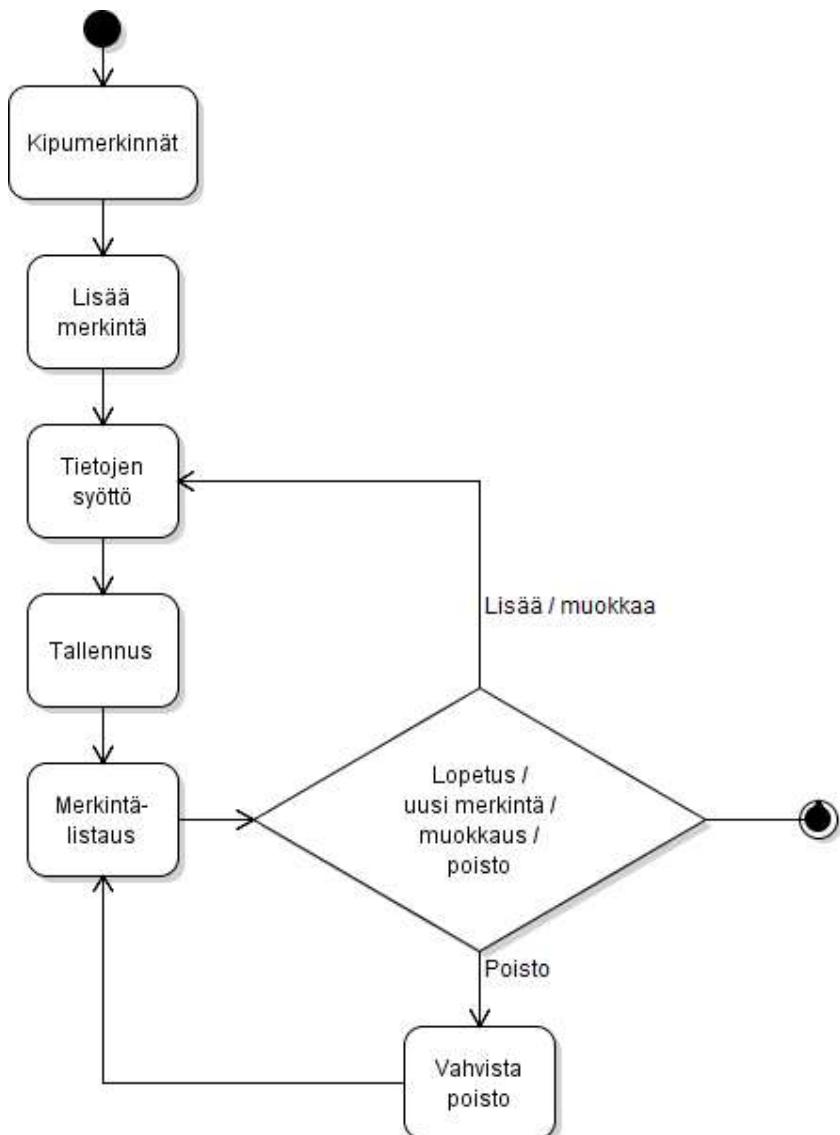
3.3 Hyväksymistestaus

Testauksen perusteella sovellus vastaa määrittelykuvastossa määritettyjä ja toteutettavaksi valittuja käyttötapauskuvauksia eikä siinä ole olennaisia toimintaan vaikuttavia puutteita.

3.3.1 Use case 1: Kipumerkinnän tallennus, muokkaus ja poisto

Testataan, onnistuuko kipumerkinnän tallennus määrittelykuvaston käyttötapaus 1:n kuvaksen mukaisesti.

Kuvaus: Käyttäjä haluaa tallentaa merkinnän kivusta. Hänen valitsee Kipumerkinnät-sivulta "Lisää merkintä". Lisättyään tarpeelliset tiedot hän painaa "Tallenna". Järjestelmä siirryy automaattisesti merkintälistaukseen, jossa mm. kiven päivämäärä, kesto ja kiven intensiteetti. Käyttäjä voi halutessaan lisätä näkymästä uuden kipumerkinnän tai muokata tai poistaa aiempia merkintöjä.

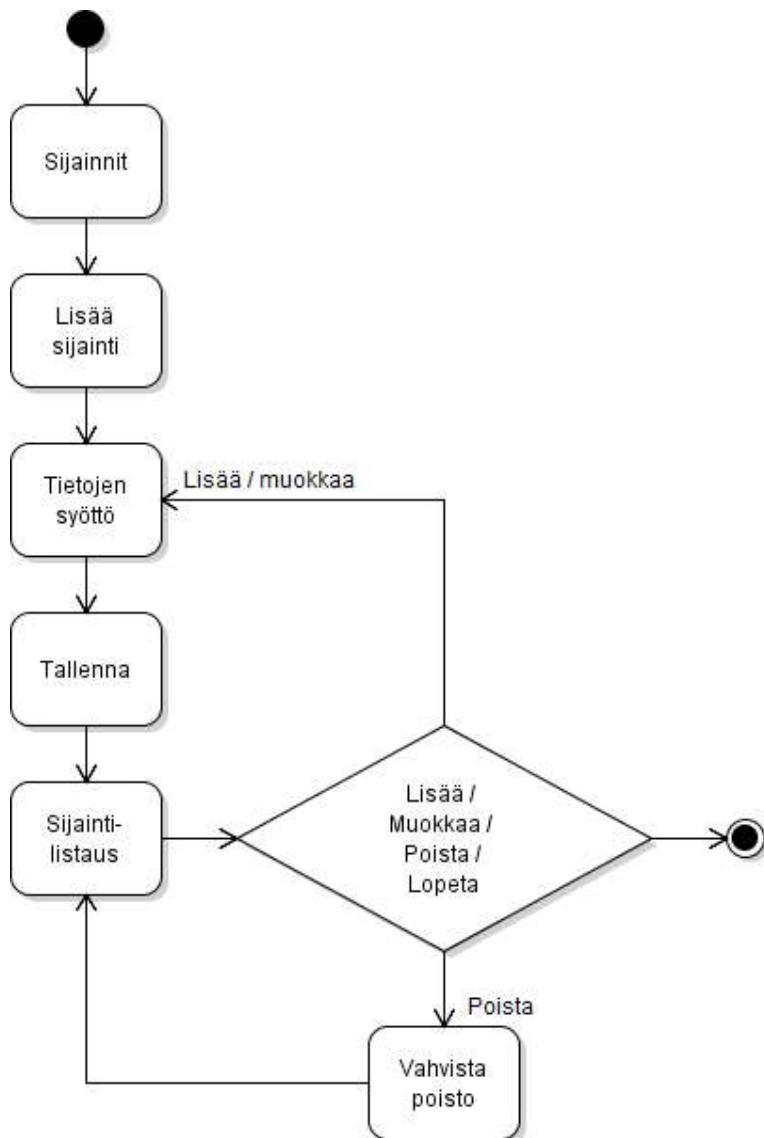


Odottettu tulos: Tiedon tallennus onnistuu olennaisilta osiltaan käyttötapauskuvaksen mukaisesti.

3.3.2 Use case 2: Kiven sijainnin lisäys

Testataan, onnistuuko sijainnin tallennus määrittelykuvaston käyttötapaus 1:n kuvaksen mukaisesti.

Kuvaus: Käyttäjä haluaa lisätä kivulle uuden sijainnin. Hän valitsee Sijainti-sivulta "Lisää sijainti" ja syöttää sijainnin nimen. Kun käyttäjä painaa tallenna, tieto siirtyy tietokantaan ja sovellus siirtyy sijaintien listaukseen. Käyttäjä voi lisätä uuden sijainnin, muokata tai poistaa aiemmin lisättyjä sijainteja tai lopettaa sovelluksen käytön.

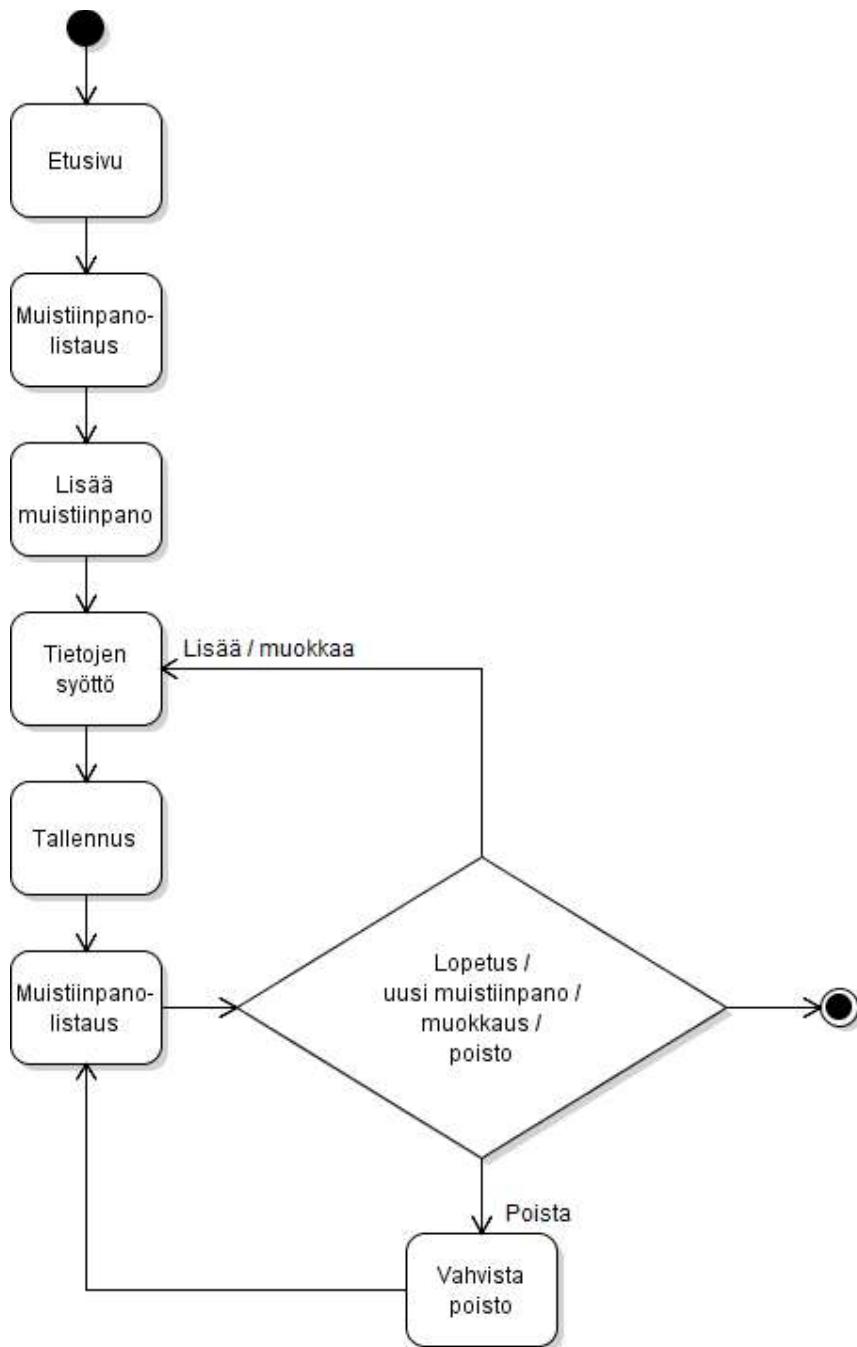


Odotettu tulos: Tiedon tallennus onnistuu olennaisilta osiltaan käyttötapauskuvauksen mukaisesti.

3.3.3 Use case 3: Erillisen muistiinpanon lisäys

Testataan, onnistuuko sijainnin tallennus määrittelykuvaston käyttötapaus 1:n kuvaukseen mukaisesti.

Kuvaus: Käyttäjä haluaa lisätä uuden muistiinpanon. Hän siirtyy muistiinpanot -sivulle ja valitsee "Lisää", syöttää muistiinpanon tiedot ja tallentaa.

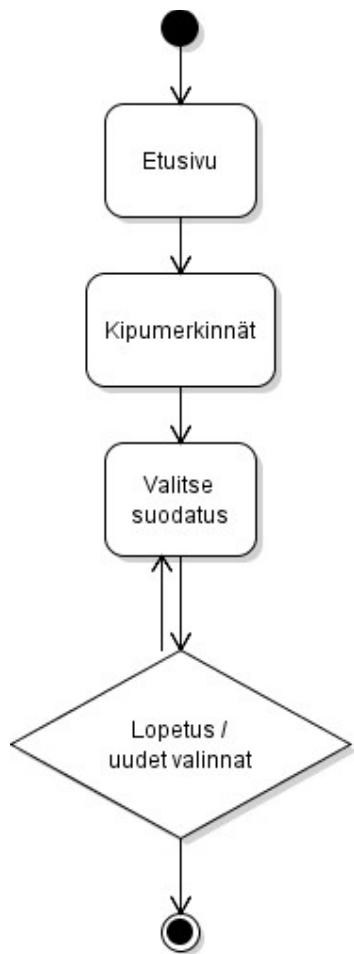


Odotettu tulos: Tiedon tallennus onnistuu olennaisilta osiltaan käyttötapauskuvauksen mukaisesti.

3.3.4 Use case 4: Kipumerkintöjen suodatus

Testataan, onnistuuko sijainnin tallennus määrittelykuvaston käyttötapaus 1:n kuvaukseen mukaisesti.

Kuvaus: Käyttäjä haluaa katsoa kipuraporttia. Hän valitsee valikosta haluamansa ajan (kuusi kuukautta, kuukausi) ja/tai kivun sijainnin. Sovellus näytää tehdyt merkinnät halutulta ajalta ja/tai sijainnista. Käyttäjä voi valita uuden suodatuksen tai lopettaa käytön.



Odottettu tulos: Tietojen suodattaminen onnistuu joko aikaa tai sijaintia tai molempia suodattamalla.

4 Vanhat tekstiosiot

5 Liitteet

Testauspöytäkirja