Testspecifikation

Systemtester

PHP kör automatiska tester. Det har varit svårt att skriva automatiska tester för de andra språken då det har tagit för mycket tid att ändra i koden. Så för de andra språken är det manuella tester som gäller.

TFS1 NODEJS

TFS1.1 - GetAllCities

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng.

Testprocedur:

Testaren besöker http://localhost:8888/GetAllCities. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF1.1: GetAllCities

1. Besök http://localhost:8888/GetAllCities.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS1.2 - GetAllCitiesWhere

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng. JSON strängen skall vara avsevärt kortare än i TFS1.1.

Testprocedur:

Testaren besöker http://localhost:8888/GetAllCitiesWhere. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF1.2: GetAllCitiesWhere

1. Besök http://localhost:8888/GetAllCitiesWhere.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS1.3 - ReadFile

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Fil Läst”.

Testprocedur:

Testaren besöker http://localhost:8888/ReadFile. Systemet skall returnera ”Fil Läst”

TF1.3: ReadFile

1. Besök http://localhost:8888/GetAllCities.
2. Systemet presenterar en ”Fil läst”.

TFS1.3.2 - ReadFile

Testplan:

Testaren försöker läsa in en fil som inte finns. Systemet skall presentera ett felmeddelande.

Testprocedur:

Byt namn på filen ”exjobb.json” i node.js mappen till ”exxjobb.json”.
Testaren besöker http://localhost:8888/ReadFile. Systemet skall returnera ”Fil Läst”

TF1.3.2: ReadFile

1. Besök http://localhost:8888/ReadFile.
2. Systemet skall presentera följande felmeddelande i consolen: { [Error: ENOENT, open 'exjobb.json'] errno: 34, code: 'ENOENT', path: 'exjobb.json' }

TFS1.4 - ReadAndSaveNew

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Fil Läst och sparad” och sparar en ny fil i mappen med namn exjobb2.json där ”_id” är ersatt av ”id”.

Testprocedur:

Testaren besöker http://localhost:8888/ReadFile. Systemet skall returnera ”Fil Läst”
Testaren öppnar exjobb.json och kontrollerar att ”_id” fältet existerar.
Testaren besöker sedan http://localhost:8888/ReadAndSaveNew. Systemet skapar då upp exjobb2.json. Testaren öppnar exjobb2.json och kontrollerar att ”_id” är ersatt av ”id”.

TF1.4: ReadAndSaveNew

1. Öppna exjobb.json, kontrollera att ”_id” fältet existerar
2. Besök http://localhost:8888/ReadAndSaveNew.
3. Systemet skapar exjobb2.json
4. Öppna exjobb2.json och kontrollera att ”_id” är ersatt av ”id”.

TFS1.5 - CalculateModulus

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Modulus Done: 3333334”.

Testprocedur:

Testaren besöker http://localhost:8888/CalculateModulus. Systemet skall returnera ” Modulus Done: 3333334”

TF1.5: CalculateModulus

1. Besök http://localhost:8888/CalculateModulus.
2. Systemet returnerar ”Modulus Done: 3333334”

TFS1.6 - SelectAndUpdate

Testplan:

Testaren kontrollerar att namnet på city har ändrat storlek, antingen till upper eller lowercase. Testaren kontrollerar namnet, besöker angiven URL och kontrollerar sedan att namnet ändrat storlek.

Testprocedur:

Testaren startar upp en ny Mongoconsoll med hjälp av commandot mongo.
Testaren väljer att använda exjobb-databasen med hjälp av ”use exjobb”
Testaren kör följande query: ”db.cities.find({ population: { $lt: 10000 } })”
Notera om städerna är skrivna med små eller stora bokstäver.
Besök angiven URL, kör samma query igen och notera om de har ändrat storlek.

TF1.6: SelectAndUpdate

1. Starta mongo consoll, kör kommandot ”use exjobb”
2. Kör queryn ”db.cities.find({ population: { $lt: 10000 } })”, notera om city är angivet i upper eller lowercase.
3. Besök http://localhost:8888/SelectAndUpdate
4. Systemet presenterar ” 21825 rows selected and updated”
5. Kör queryn ” db.cities.find({ population: { $lt: 10000 } })” och kontrollera om storleken har ändrats

TFS2 PHP

Automatiska tester. Testaren ändrar i service.php filen, tar bort kommentarerna längst ner i varje funktion så att funktionerna returnerar istället för ”echo”.
Besök sedan localhost/exjobb/Test.
Kontrollera att varje test är success.

TFS3 DJANGO

TFS3.1 - GetAllCities

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng.

Testprocedur:

Testaren besöker http://localhost:3000/GetAllCities/. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF3.1: GetAllCities

1. Besök http://localhost:3000/GetAllCities/.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS3.2 - GetAllCitiesWhere

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng. JSON strängen skall vara avsevärt kortare än i TFS1.1.

Testprocedur:

Testaren besöker http://localhost:3000/GetAllCitiesWhere/. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF3.2: GetAllCitiesWhere

1. Besök http://localhost:3000/GetAllCitiesWhere/.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS3.3 - ReadFile

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Fil Läst”.

Testprocedur:

Testaren besöker http://localhost:3000/ReadFile/. Systemet skall returnera ”Fil Läst”

TF3.3: ReadFile

1. Besök http://localhost:3000/GetAllCities/.
2. Systemet presenterar en ”Fil läst”.

TFS3.4 - ReadAndSaveNew

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Fil Läst och sparad” och sparar en ny fil i mappen med namn exjobb2.json där ”_id” är ersatt av ”id”.

Testprocedur:

Testaren besöker http://localhost:3000/ReadFile/. Systemet skall returnera ”Fil Läst”
Testaren öppnar exjobb.json och kontrollerar att ”_id” fältet existerar.
Testaren besöker sedan http://localhost:3000/ReadAndSaveNew/. Systemet skapar då upp exjobb2.json. Testaren öppnar exjobb2.json och kontrollerar att ”_id” är ersatt av ”id”.

TF3.4: ReadAndSaveNew

1. Öppna exjobb.json, kontrollera att ”_id” fältet existerar
2. Besök http://localhost:3000/ReadAndSaveNew/.
3. Systemet skapar exjobb2.json
4. Öppna exjobb2.json och kontrollera att ”_id” är ersatt av ”id”.

TFS3.5 - CalculateModulus

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Modulus Done: 3333334”.

Testprocedur:

Testaren besöker http://localhost:3000/CalculateModulus. Systemet skall returnera ”Modulus Done: 3333334”

TF3.5: CalculateModulus

1. Besök http://localhost:3000/CalculateModulus/.
2. Systemet returnerar ”Modulus Done”

TFS3.6 - SelectAndUpdate

Testplan:

Testaren kontrollerar att namnet på city har ändrat storlek, antingen till upper eller lowercase. Testaren kontrollerar namnet, besöker angiven URL och kontrollerar sedan att namnet ändrat storlek.

Testprocedur:

Testaren startar upp en ny Mongoconsoll med hjälp av commandot mongo.
Testaren väljer att använda exjobb-databasen med hjälp av ”use exjobb”
Testaren kör följande query: ”db.cities.find({ population: { $lt: 10000 } })”
Notera om städerna är skrivna med små eller stora bokstäver.
Besök angiven URL, kör samma query igen och notera om de har ändrat storlek.

TF3.6: SelectAndUpdate

1. Starta mongo consoll, kör kommandot ”use exjobb”
2. Kör queryn ”db.cities.find({ population: { $lt: 10000 } })”, notera om city är angivet i upper eller lowercase.
3. Besök http://localhost:3000/SelectAndUpdate/
4. Systemet presenterar ” 21825 rows selected and updated”
5. Kör queryn ” db.cities.find({ population: { $lt: 10000 } })” och kontrollera om storleken har ändrats

TFS4 RAILS

TFS4.1 - GetAllCities

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng.

Testprocedur:

Testaren besöker http://localhost:3000/home/GetAllCities. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF4.1: GetAllCities

1. Besök http://localhost:3000/home/GetAllCities.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS4.2 - GetAllCitiesWhere

Testplan:

Testaren besöker angiven URL och systemet returnerar all data ifrån databasen iform av en JSON sträng. JSON strängen skall vara avsevärt kortare än i TFS1.1.

Testprocedur:

Testaren besöker http://localhost:3000/home/GetAllCitiesWhere. Systemet skall returnera en JSON Sträng där första strängen skall ha ”Adamsville” som ”city” och den sista skall ha ”Logan”.

TF4.2: GetAllCitiesWhere

1. Besök http://localhost:3000/home/GetAllCitiesWhere.
2. Systemet presenterar en JSON - Sträng
3. Kontrollera att första objektet innehåller ”Adamsville” som city och sista objektet innehåller ”Logan”

TFS4.3 - ReadFile

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Läst klart”

Testprocedur:

Testaren besöker http://localhost:3000/home/ReadFile. Systemet skall returnera ”Fil Läst”

TF1.3: ReadFile

1. Besök http://localhost:3000/home/ReadFile.
2. Systemet presenterar ”Läst klart”.

TFS4.3.2 - ReadFile

Testplan:

Testaren försöker läsa in en fil som inte finns. Systemet skall presentera ett felmeddelande.

Testprocedur:

Byt namn på filen ”exjobb.json” i node.js mappen till ”exxjobb.json”.
Testaren besöker http://localhost:3000/home/ReadFile. Systemet skall presentera ett felmeddelande

TF4.3.2: ReadFile

1. Besök http://localhost:3000/home/ReadFile.
2. Systemet skall presentera följande felmeddelande ”`No such file or directory - exjobb.json`”

TFS4.4 - ReadAndSaveNew

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Fil Läst och sparad” och sparar en ny fil i mappen med namn exjobb2.json där ”_id” är ersatt av ”id”.

Testprocedur:

Testaren besöker http://localhost:3000/home/ReadAndSaveNew. Systemet skall returnera ”Läst och Modifierat”
Testaren öppnar exjobb.json och kontrollerar att ”_id” fältet existerar.
Testaren besöker sedan http://localhost:3000/home/ReadAndSaveNew Systemet skapar då upp exjobb2.json. Testaren öppnar exjobb2.json och kontrollerar att ”_id” är ersatt av ”id”.

TF4.4: ReadAndSaveNew

1. Öppna exjobb.json, kontrollera att ”_id” fältet existerar
2. Besök http://localhost:8888/ReadAndSaveNew.

3. Systemet skapar exjobb2.json
4. Öppna exjobb2.json och kontrollera att ”_id” är ersatt av ”id”.

TFS4.5 - CalculateModulus

Testplan:

Testaren besöker angiven URL och systemet returnerar ”Modulus klar”.

Testprocedur:

Testaren besöker http://localhost:8888/CalculateModulus. Systemet skall returnera ” Modulus Done”

TF4.5: CalculateModulus

1. Besök http://localhost:3000/home/CalculateModulus.
2. Systemet returnerar ”Modulus Done”

TFS4.6 - SelectAndUpdate

Testplan:

Testaren kontrollerar att namnet på city har ändrat storlek, antingen till upper eller lowercase. Testaren kontrollerar namnet, besöker angiven URL och kontrollerar sedan att namnet ändrat storlek.

Testprocedur:

Testaren startar upp en ny Mongoconsoll med hjälp av commandot mongo.
Testaren väljer att använda exjobb-databasen med hjälp av ”use exjobb”
Testaren kör följande query: ”db.cities.find({ population: { $lt: 10000 } })”
Notera om städerna är skrivna med små eller stora bokstäver.
Besök angiven URL, kör samma query igen och notera om de har ändrat storlek.

TF4.6: SelectAndUpdate

1. Starta mongo consoll, kör kommandot ”use exjobb”
2. Kör queryn ”db.cities.find({ population: { $lt: 10000 } })”, notera om city är angivet i upper eller lowercase.
3. Besök http://localhost:3000/home/SelectAndUpdate
4. Systemet presenterar ” Selected and updated done”
5. Kör queryn ” db.cities.find({ population: { $lt: 10000 } })” och kontrollera om storleken har ändrats