Høgskulen på Vestlandet

Fakultet for ingeniør- og naturvitskap

Institutt for datateknologi, elektroteknologi og realfag

Automatisert klienttesting
Systemdokumentasjon

Versjon 1.0

Dokumentet er basert på Systemdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
24/04/23	1.0	La til kapittel 1, 3,4,5	Jan William Jensen
11/05/23	2.0	Fikset kapittel 2,3,4,5	Jan William Jensen
21/05/23	3.0	Lagt til bildene fra Swagger doc. Skrevet i kapittel 9.	Jan William Jensen

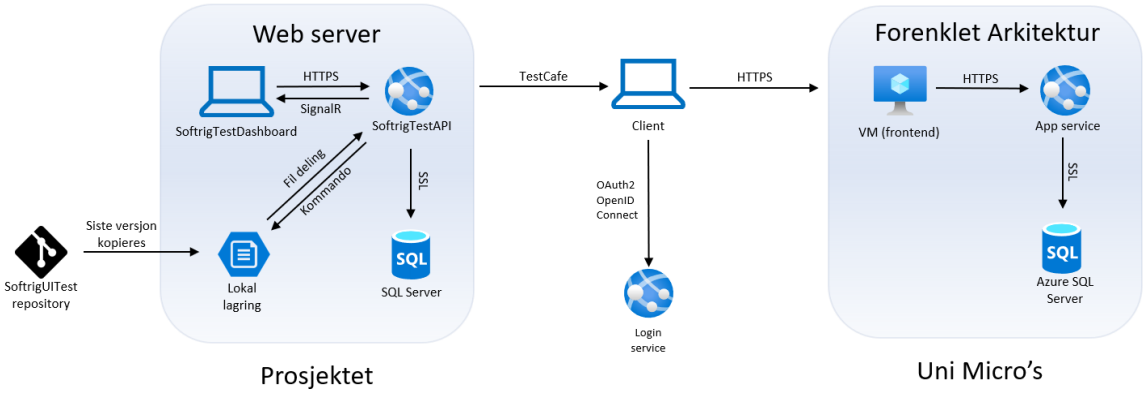
INNHOLDSFORTEGNELSE

1 INNLEDNING

Dette dokumentet er skrevet i sammenheng med bacheloroppgave ved Høgskulen på Vestlandet vårsemesteret 2023. Dokumentet har som hensikt å gi en kort innføring i systemet som er utviklet i prosjektet. Dokumentet forklarer systemets arkitektur, prosjektstruktur, og tekniske detaljer.

2 ARKITEKTUR

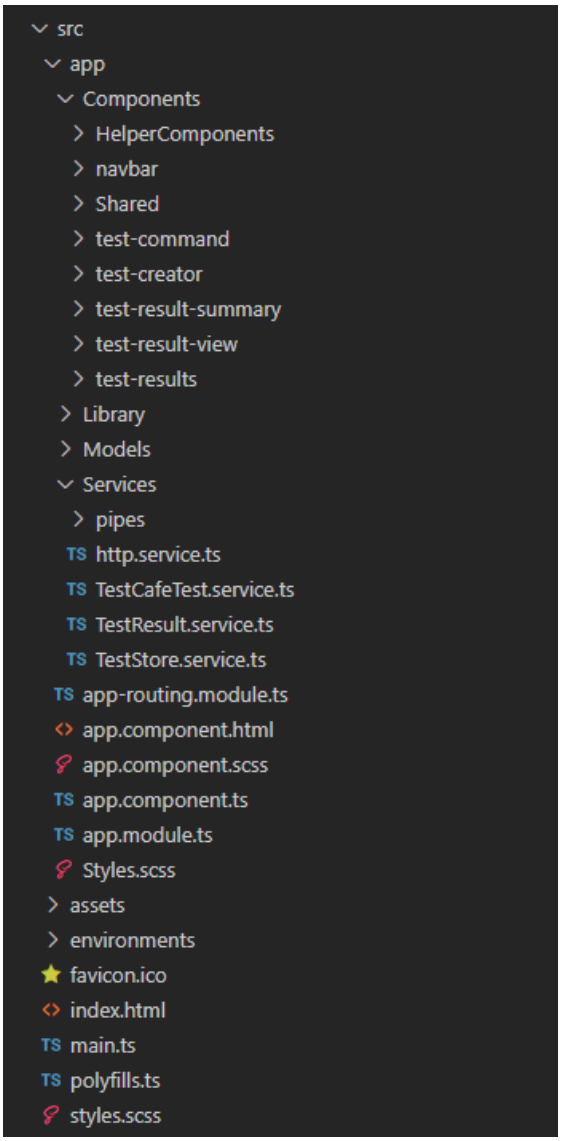
Utviklet løsning har et brukergrensesnitt og et programmeringsgrensesnitt. Brukergrensesnittet sender HTTPS-kall til programmeringsgrensesnittet som er basert på REST. All data blir lagret i en SQL Server og testene er lagret som et separat repository.



Figur 1: Arkitekturskisse. SoftrigUITest er sendt inn av Uni Micro til en lokal mappe som APIet leser fra. Testene som startes utfører instruksjonene som er spesifisert i testene.

3 PROSJEKTSTRUKTUR

Vis fil- og katalogstrukturen for prosjektet; kildekode, biblioteker, eksekverbar kode og hvordan disse er organisert. Løsningen er delt inn i tre prosjekter, en Angular-applikasjon og to .NET prosjekter (API og Tester). Inne i APIet er det igjen delt inn i mapper ut i fra ansvarsområder: Controller, Service, Model, Database, Migrations.

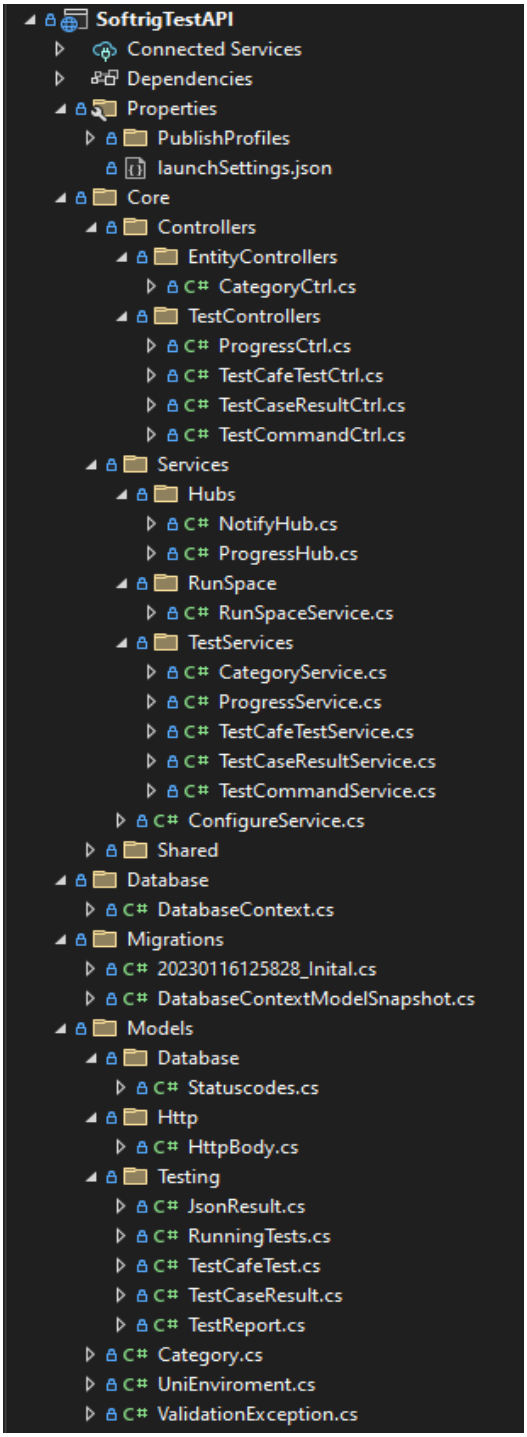


Figur 2: Illustrasjon av filstrukturen til SoftrigTestDashboard

Dashbordet er ansvarlig for brukergrensesnittet og inneholder følgende mapper:

- **App** - Inneholder alle komponentene som brukes i prosjektet. Denne mappen består igjen av
    - **Components** - alle komponentene som bygger opp sidene til applikasjonen.
    - **Library** - Konstanter.
    - **Models** - Alle definisjonene på objekter som sendes til og fra appen.
    - **Services** - tjenester som gjennomfører ulik funksjonalitet.
- **Assets** - Inneholder bilder og SVG filer for piler og andre ting

- **Environments** - inneholder url adresser til ulike miljøer
- **Styles** - inneholder kode for utseende til appen. Alle komponentene har egne styling filer som arver fra scss filer lengre opp i hierarkiet



Figur 3: Illustrasjon av filstrukturen til SoftrigTestAPI

APIet er ansvarlig for oppstart av tester, generering og formatering av resultater og kommunikasjon med databasen og inneholder følgende mapper:

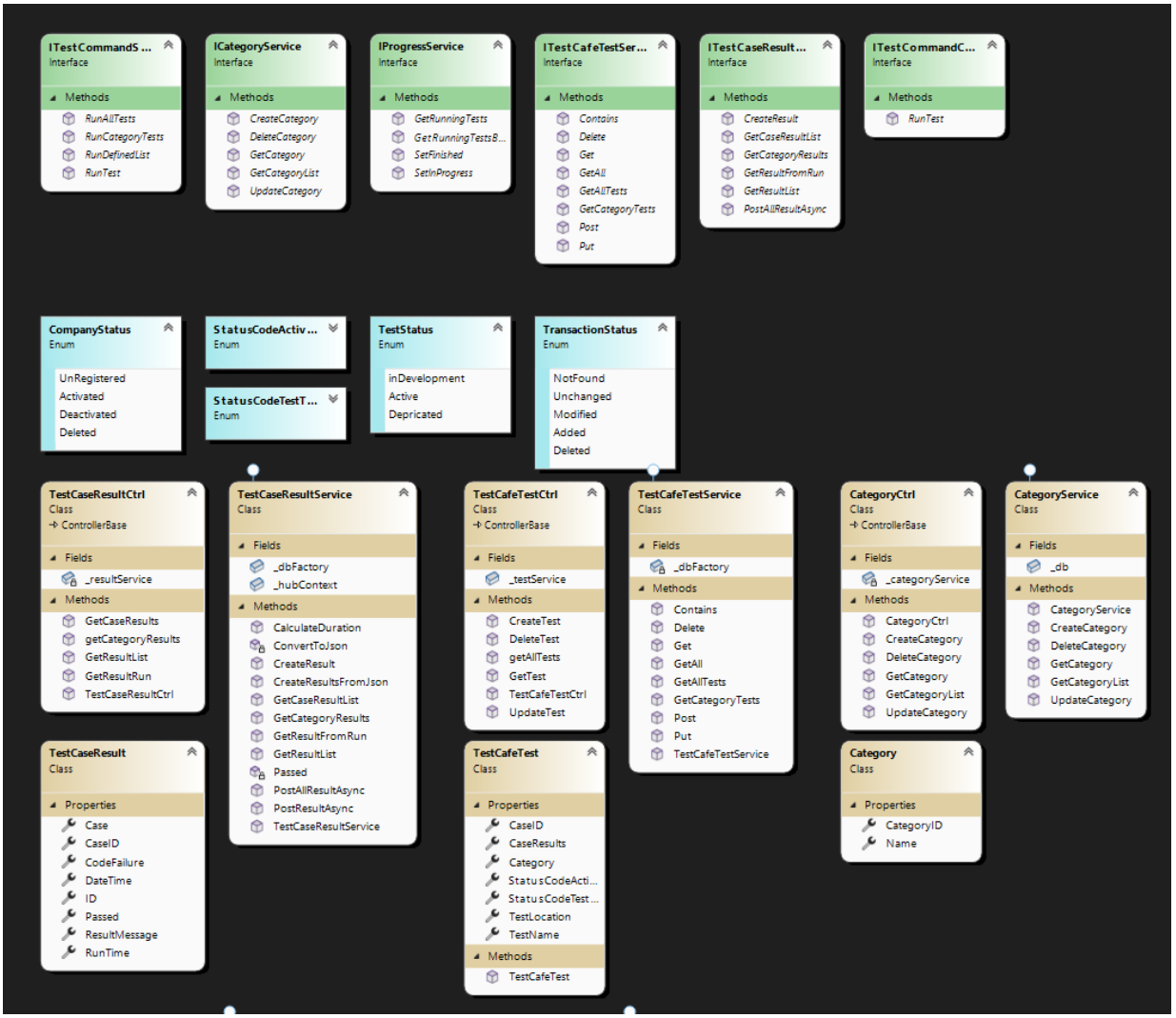
- **Core** - Kjernefunksjonaliteten i prosjektet. Videre delt inn i Controllers og Services
    - **Controller** - inneholder alle endepunktene til APIet

    - **Services** - inneholder kobling mot database for respektive ressurser og kjøring av tester.
- **Database** - database kontekst som inneholder connection string
- **Migrations** - inneholder kode for å generere og oppdatere database ut i fra koden
- **Models** - inneholder objekt definisjoner som brukes i APIet og som sendes til klientene.

4 KLASSEDIAGRAM

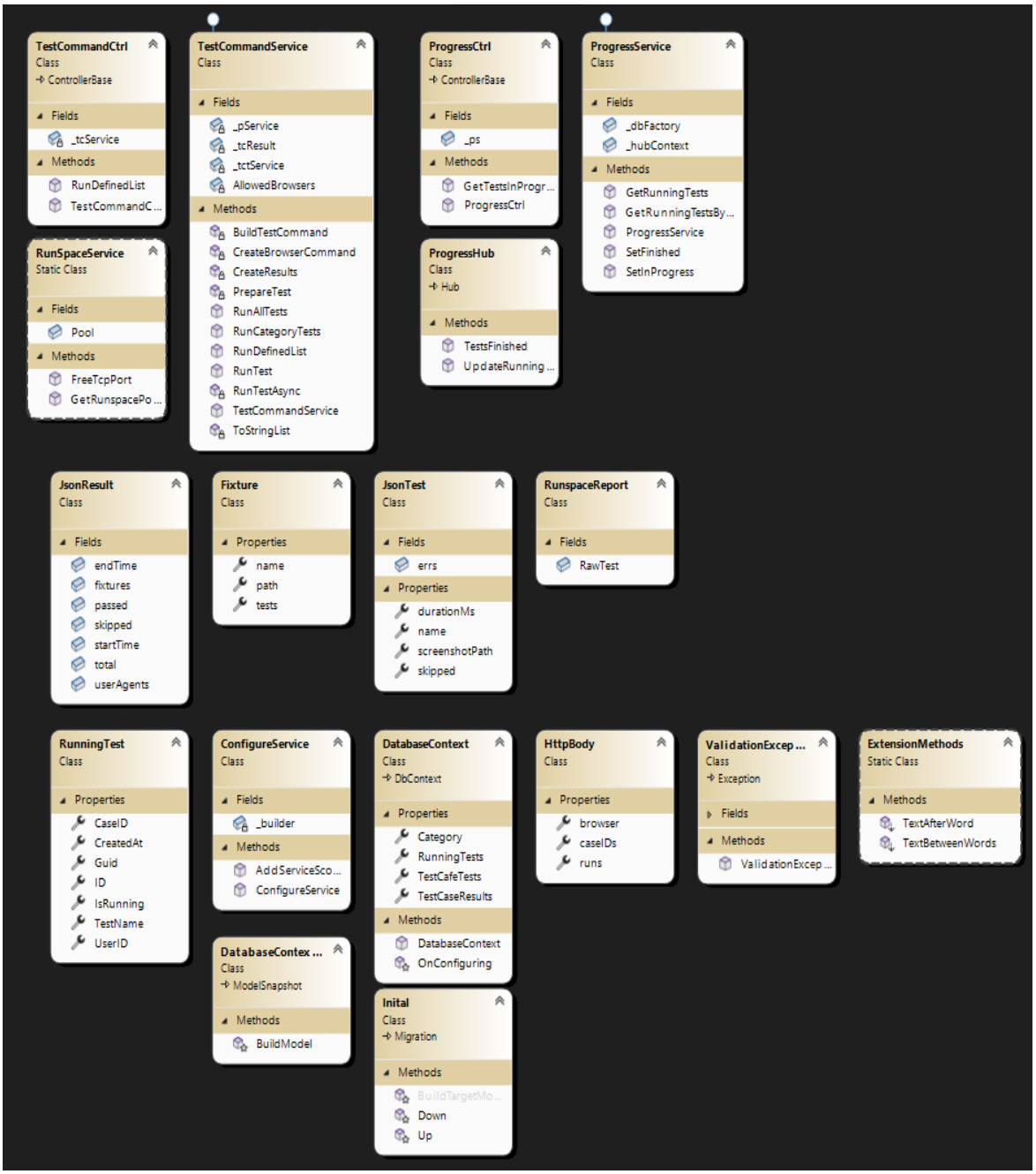
Klassediagram viser ulike interface grønt, Enums i blått og klasser i gult. De gule klassene er igjen delt inn og sortert etter TestCaseResult, TestCafeTest, Category, TestCommand og Progress (SignalR).

Alle kontrollere arver fra ControllerBase og dirigerer videre til service klasser for å utføre handlingene mot databasen og oppstart av tester.

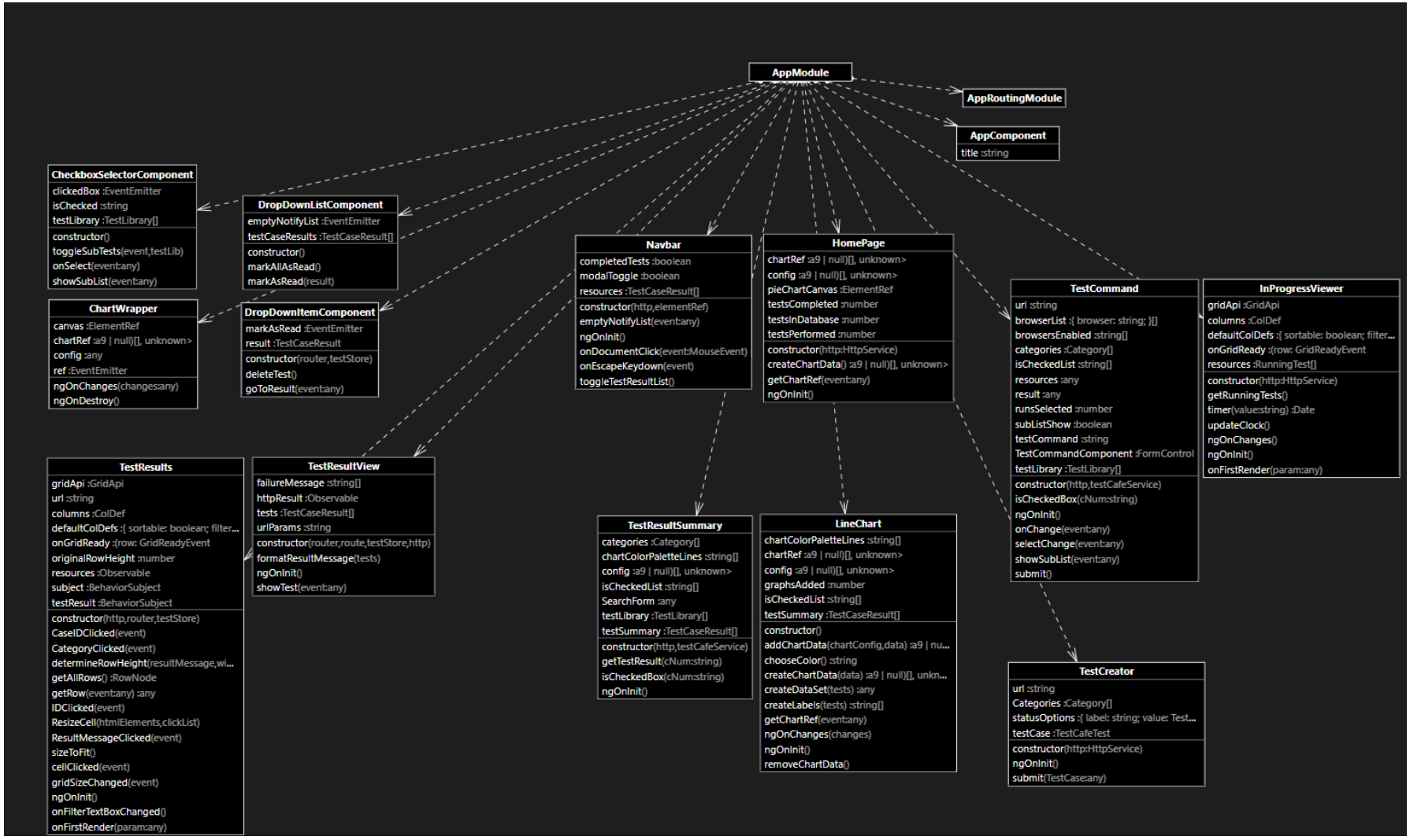


Figur 4: Del 1 av klassediagram med attributter og metoder.

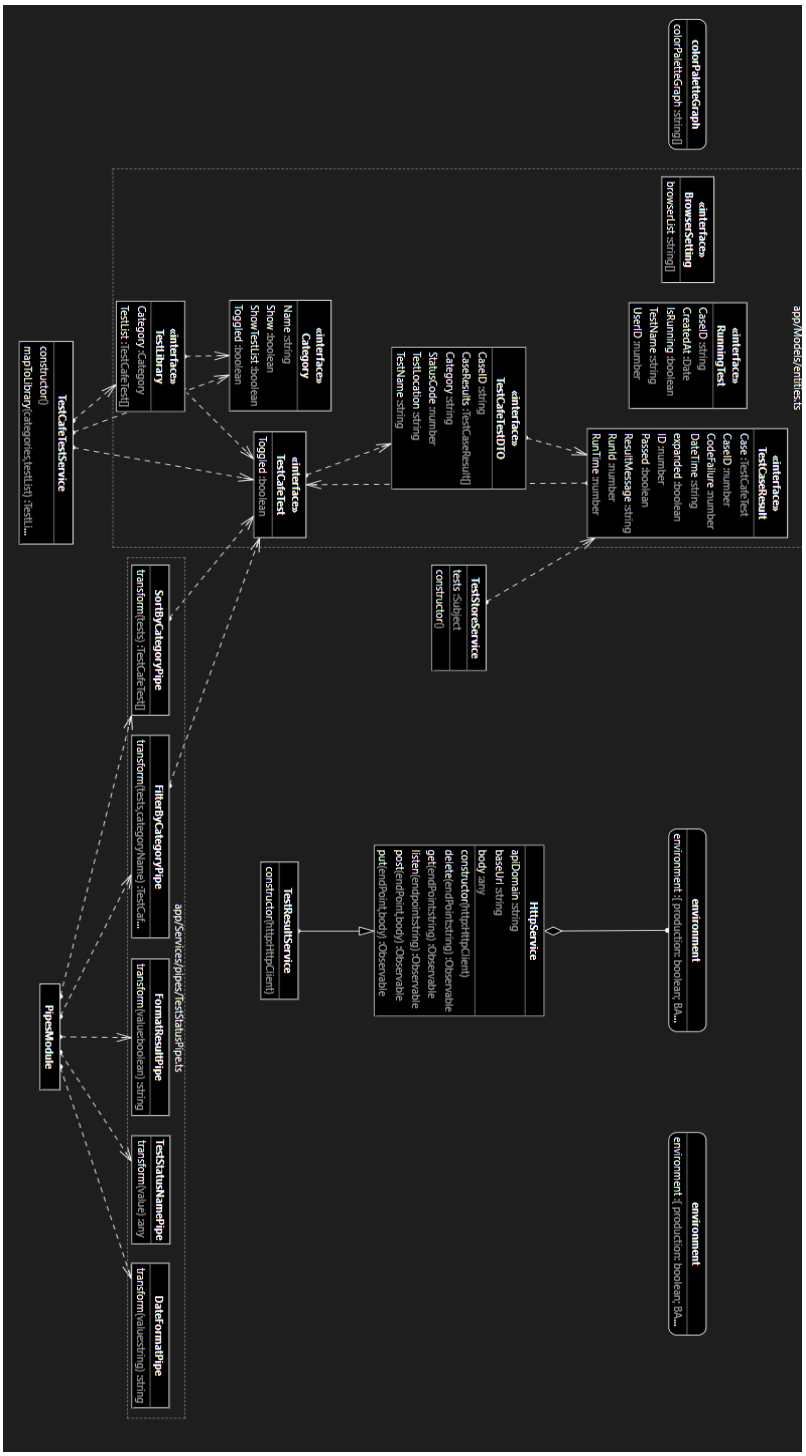
I figur 5 er det klasser med navn JsonResult, Fixture, JsonTest og RunspaceReport som benyttes for å hjelpe JSON å konvertere fra tekst til objekter som til slutt blir TestCaseResult.



Figur 5: Del 2 av klassediagram med attributter og metoder.

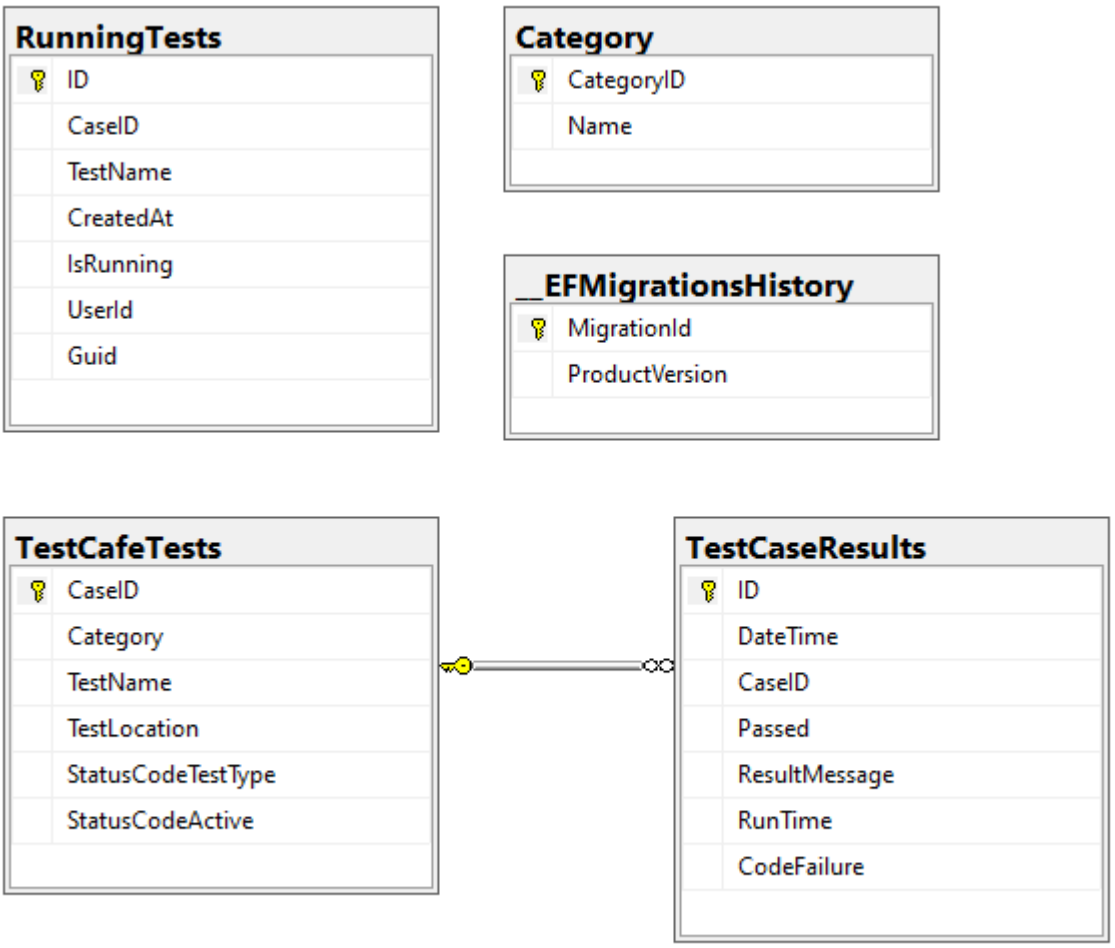

AppModule
AppRoutingModule
AppComponent
title :string
CheckboxSelectorComponent
clickedBox :EventEmitter
isChecked :string
testLibrary :TestLibrary[]
constructor()
toggleSubTests(event,testLib)
onSelect(event:any)
showSubList(event:any)
DropDownListComponent
emptyNotifyList :EventEmitter
testCaseResults :TestCaseResult[]
constructor()
markAllAsRead()
markAsRead(result)
ChartWrapper
canvas :ElementRef
chartRef :a9 | null)[], unknown>
config :any
ref :EventEmitter
ngOnChanges(changes:any)
ngOnDestroy()
DropDownItemComponent
markAsRead :EventEmitter
result :TestCaseResult
constructor(router,testStore)
deleteTest()
goToResult(event:any)
Navbar
completedTests :boolean
modalToggle :boolean
resources :TestCaseResult[]
constructor(http,elementRef)
emptyNotifyList(event:any)
ngOnInit()
onDocumentClick(event:MouseEvent)
onEscapeKeydown(event)
toggleTestResultList()
HomePage
chartRef :a9 | null)[], unknown>
config :a9 | null)[], unknown>
pieChartCanvas :ElementRef
testsCompleted :number
testsInDatabase :number
testsPerformed :number
constructor(http:HttpService)
createChartData() :a9 | null)[], unknown>
getChartRef(event:any)
ngOnInit()
TestCommand
url :string
browserList :{ browser: string; }[]
browsersEnabled :string[]
categories :Category[]
isCheckedList :string[]
resources :any
result :any
runsSelected :number
subListShow :boolean
testCommand :string
TestCommandComponent :FormControl
testLibrary :TestLibrary[]
constructor(http,testCafeService)
isCheckedBox(cNum:string)
ngOnInit()
onChange(event:any)
selectChange(event:any)
showSubList(event:any)
submit()
InProgressViewer
gridApi :GridApi
columns :ColDef
defaultColDefs :{ sortable: boolean; filter...
onGridReady :(row: GridReadyEvent
resources :RunningTest[]
constructor(http:HttpService)
getRunningTests()
timer(value:string) :Date
updateClock()
ngOnChanges()
ngOnInit()
onFirstRender(param:any)
TestResults
gridApi :GridApi
url :string
columns :ColDef
defaultColDefs :{ sortable: boolean; filter...
onGridReady :(row: GridReadyEvent
originalRowHeight :number
resources :Observable
subject :BehaviorSubject
testResult :BehaviorSubject
constructor(http,router,testStore)
CaseIDClicked(event)
CategoryClicked(event)
determineRowHeight(resultMessage,wi...
getAllRows() :RowNode
getRow(event:any) :any
IDClicked(event)
ResizeCell(htmlElements,clickList)
ResultMessageClicked(event)
sizeToFit()
cellClicked(event)
gridSizeChanged(event)
ngOnInit()
onFilterTextBoxChanged()
onFirstRender(param:any)
TestResultView
failureMessage :string[]
httpResult :Observable
tests :TestCaseResult[]
urlParams :string
constructor(router,route,testStore,http)
formatResultMessage(tests)
ngOnInit()
showTest(event:any)
TestResultSummary
categories :Category[]
chartColorPaletteLines :string[]
config :a9 | null)[], unknown>
isCheckedList :string[]
SearchForm :any
testLibrary :TestLibrary[]
testSummary :TestCaseResult[]
constructor(http,testCafeService)
getTestResult(cNum:string)
isCheckedBox(cNum:string)
ngOnInit()
LineChart
chartColorPaletteLines :string[]
chartRef :a9 | null)[], unknown>
config :a9 | null)[], unknown>
graphsAdded :number
isCheckedList :string[]
testSummary :TestCaseResult[]
constructor()
addChartData(chartConfig,data) :a9 | nu...
chooseColor() :string
createChartData(data) :a9 | null)[], unkn...
createDataSet(tests) :any
createLabels(tests) :string[]
getChartRef(event:any)
ngOnChanges(changes)
ngOnInit()
removeChartData()
TestCreator
url :string
Categories :Category[]
statusOptions :[label: string; value: Test...
testCase :TestCafeTest
constructor(http:HttpService)
ngOnInit()
submit(TestCase:any)

I øvre del ser man AppModule og hvordan den er koblet med alle komponenter. I nedre del er kobling mellom PipesModule som er importert i AppModule og alle objektdefinisjoner



Link til nedlastbar fullskala versjon: [Fullskala versjon]

5 DATABASEMODELL



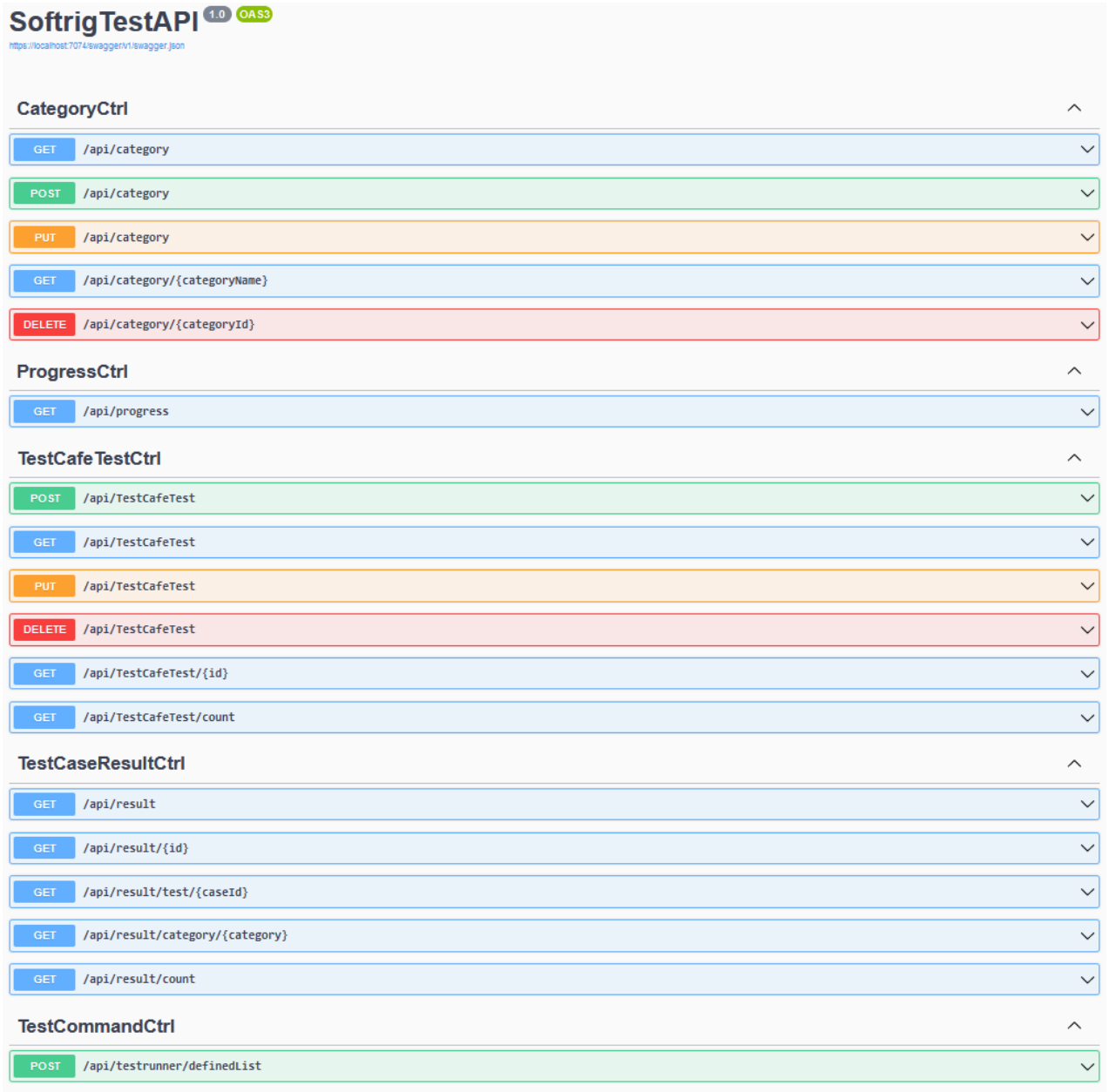
Databasen inneholder følgende tabeller:

- **TestCafeTests** - alle testene.
- **TestCaseResults** - alle resultatene.
- **RunningTests** - RunningTests inneholder tester som kjører.
- **Category** - er en tabell med alle kategorier av tester. Denne er nødvendig så lenge vi ikke er koblet mot TestRail.
- **EFMigrationsHistory** - EFMigrationsHistory er en logg som brukes når man gjør endringer i databasen. Da vil de forskjellige migrasjons filene være spesifisert i denne tabellen slik at alle som skal bruke programmet vil gå gjennom de samme migrasjonene og får en identisk database.

6 Server-tjenester

- **CategoryCtrl** er endepunktene for kategorien til en test
- **ProgressCtrl** er endepunktet for tester som kjører. Denne ressursen blir også gitt til klientene gjennom WebSocket
- **TestCafeTestCtrl** er endepunktene for alle testene. Man kan også hente antall resultater
- **TestCaseResultCtrl** er endepunktene for alle resultatene. Her finnes det forskjellige endepunkter som henter resultater basert på ID, C-nummer og kategori. Man kan også hente antall resultater
- **TestCommandCtrl** er endepunktet som starter testene.

- Det finnes også to WebSocket endepunkter.
    - **UpdateRunningTest** - sender beskjed til klientene om hvilken tester som kjører.
    - **TestsFinished**. - ferdige resultater til klientene slik at brukeren får en notifikasjon med ferdige resultater.


SoftrigTestAPI 1.0 OAS3
https://localhost:7074/swagger/v1/swagger.json
CategoryCtrl
GET /api/category
POST /api/category
PUT /api/category
GET /api/category/{categoryName}
DELETE /api/category/{categoryId}
ProgressCtrl
GET /api/progress
TestCafeTestCtrl
POST /api/TestCafeTest
GET /api/TestCafeTest
PUT /api/TestCafeTest
DELETE /api/TestCafeTest
GET /api/TestCafeTest/{id}
GET /api/TestCafeTest/count
TestCaseResultCtrl
GET /api/result
GET /api/result/{id}
GET /api/result/test/{caseId}
GET /api/result/category/{category}
GET /api/result/count
TestCommandCtrl
POST /api/testrunner/definedList

7 SIKKERHET

Applikasjonen vil ikke være tilgjengelig på internett. Denne applikasjonen er kun tilgjengelig på Uni Micro's intranett, og det er derfor ikke tatt hensyn til sikkerhet utover det som er innebygd i .NET Core og Entity Framework Core

8 INSTALLASJON OG KJØRING

Backend er avhengig av:

- Microsoft.NET.SDK - er en utviklingspakke som inneholder funksjonalitet for .NET Core.
- Visual Studio 2022 - utviklingsverktøyet Backend er skrevet i.

Alle tester som er skrevet av oppdragsgiver ligger i et privat repository på GitHub. Det er ikke mulig å kjøre noen tester uten tilgang til disse testene. Om testene er lastet ned på maskinen skal de legges i en mappe for siden av Backend.

For å installere og kjøre Backend må følgende gjøres:

1. Klone prosjektet fra GitHub
2. Starte prosjektets solution-fil (.sln) i Visual Studio
3. Starte prosjektet (Ctrl + F5)

Frontend er avhengig av:

- Node.js - Node inneholder all funksjonalitet som benyttes i Angular
- Typescript - et supersett av JavaScript som tilfører type sikring.
- Angular - Rammeverket Frontend er skrevet i.

For å installere og kjøre Frontend må følgende gjøres

1. Klone prosjektet fra GitHub.
2. Åpne terminalen i prosjektet
3. npm install
4. npm start

Verktøyet er tilgjengelig på https://localhost:4200

9 DOKUMENTASJON AV KILDEKODE

Koden er hovedsakelig dokumentert i koden.
APIet er dokumentert ved hjelp av swagger og JSON til PDF konvertering

Paths

GET /api/category

Responses

Code	Description	Links
200	Success	No Links

PUT /api/category

Parameters

Type	Name	Description	Schema
query	**CategoryID** *optional*		integer (int32)
query	**Name** *optional*		string

Responses

Code	Description	Links
200	Success	No Links

POST /api/category

Parameters

Type	Name	Description	Schema
query	**CategoryID** *optional*		integer (int32)
query	**Name** *optional*		string

Responses

Code	Description	Links
200	Success	No Links

GET /api/category/{categoryName}

Parameters

Type	Name	Description	Schema
path	**categoryName** *required*		string

Responses

Code	Description	Links
200	Success	No Links

DELETE /api/category/{categoryId}

Parameters

Type	Name	Description	Schema
path	**categoryId** *required*		integer (int32)

Responses

Code	Description	Links
200	Success	No Links

GET /api/progress

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

GET /api/TestCafeTest

Responses

Code	Description	Links
200	Success	No Links

PUT /api/TestCafeTest

Parameters

Type	Name	Description	Schema
query	**TestName** *optional*		string
query	**CaseResults** *optional*		< TestCaseResult > array
query	**Category** *optional*		string
query	**StatusCodeActive** *optional*		integer (int32)
query	**TestLocation** *optional*		string
query	**CaseID** *required*		string
query	**StatusCodeTestType** *optional*		integer (int32)

Responses

Code	Description	Links
200	Success	No Links

POST /api/TestCafeTest

Parameters

Type	Name	Description	Schema
query	**TestName** *optional*		string
query	**CaseResults** *optional*		< TestCaseResult > array
query	**Category** *optional*		string
query	**StatusCodeActive** *optional*		integer (int32)
query	**TestLocation** *optional*		string

Type	Name	Description	Schema
query	**CaseID** *required*		string
query	**StatusCodeTestType** *optional*		integer (int32)

Responses

Code	Description	Links
200	Success	No Links

DELETE /api/TestCafeTest

Parameters

Type	Name	Description	Schema
query	**caseID** *optional*		string

Responses

Code	Description	Links
200	Success	No Links

GET /api/TestCafeTest/{id}

Parameters

Type	Name	Description	Schema
path	**id** *required*		string

Responses

Code	Description	Links
200	Success	No Links

GET /api/TestCafeTest/count

Responses

Code	Description	Links
200	Success	No Links

GET /api/result

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

GET /api/result/{id}

Parameters

Type	Name	Description	Schema
path	**id** *required*		integer (int32)

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

GET /api/result/test/{caseId}

Parameters

Type	Name	Description	Schema
path	**caseId** *required*		string

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

GET /api/result/category/{category}

Parameters

Type	Name	Description	Schema
path	**category** *required*		string

Responses

Code	Description	Links
200	Success	No Links

GET /api/result/count

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

POST /api/testrunner/definedList

Responses

Code	Description	Links
200	Success *Content* **text/plain** **application/json** **text/json**	No Links

Components

Schemas

HttpBody

Properties

Name	Description	Schema
browser *optional*	*nullable*	< string > array
runs *optional*		integer (int32)
caseIDs *optional*	*nullable*	< string > array

RunningTest

Properties

Name	Description	Schema
ID *optional*		integer (int32)
CaseID *optional*	*nullable*	string
TestName *optional*	*nullable*	string
CreatedAt *optional*		string (date-time)
IsRunning *optional*		boolean
UserID *optional*		integer (int32)
Guid *optional*		string (uuid)

TestCafeTest

Properties

Name	Description	Schema
CaseID *required*	**Minimum Length**: 1	string
Category *optional*	*nullable*	string

Name	Description	Schema
TestName *optional*	*nullable*	string
TestLocation *optional*	*nullable*	string
StatusCodeTestType *optional*		integer (int32)
StatusCodeActive *optional*		integer (int32)
CaseResults *optional*	*nullable*	< TestCaseResult > array

TestCaseResult

Properties

Name	Description	Schema
ID *optional*		integer (int32)
DateTime *optional*		string (date-time)
CaseID *required*	**Minimum Length**: 1	string
Case *optional*		TestCafeTest
Passed *optional*		boolean
ResultMessage *optional*	*nullable*	string
RunTime *optional*		integer (int32)
CodeFailure *optional*		integer (int32)

10 KONTINUERLIG INTEGRASJON OG TESTING

Det har ikke vært gjennomført noe kontinuerlig integrasjon mot server på intranettet til Uni Micro.

Prosjektet har gjennom utvikling benyttet seg av en hovedgren med navn develop og master (henholdsvis Backend og Frontend) hvor utvikling har skjedd i grener fra disse. Når ønsket funksjonalitet er implementert blir det sendt inn en pull-forespørsel hvor den en person må se over og godkjenne. Om det skulle finnes konflikter mot develop eller master blir dette løst under pull-forespørselen.

11 REFERANSER

Ingen referanse.