Pages / ... / Completed 2 Jira links

FDD0000 Display Forecast Systems

Created by Tomas Izo, last modified by Ophelia Marott Zhang on 2018-10-12

NOTE: To use this template, draw a number for the new FDD in DTS. Save a copy of this page with the title "FDD<DTS no.> <Feature name>" in the Wiki-space where your project keeps FDDs. Then read the Template user's guide below for more information.

Overview

Feature Lead

@ Tomas Izo (tomiz)

State	Responsible	Participants in Review
RFC	@ Tomas Izo	@ Tomas Izo @ Georgi Stefanov @ Adam René Gregersen
RFI	@ Tomas Izo	@ Tomas Izo @ Georgi Stefanov @ Ophelia Marott Zhang @ Adam René Gregersen

Jira

~~CSL-11393~~ - FDD0000 Display Forecast Systems CLOSED

Content

1. Feature Statement

The purpose of this feature is to provide a basis for the Forecast Monitoring System.

This feature will enable the system to receive unit(s) data from the Forecast System and display them to the user.

2. Scope and Limitations

2.1. Scope

- Get Unit(s) information from the Forecast System
  - get new information periodically (with configurable time period)
- Show units overview in a web application
  - Units:
    - Unit is shown by its name
    - units are in a following hierarchy:
      - Region/Instalation
        - Client/System
          - Unit1
          - Unit2
          - ...
  - Web application:
    - Always shows the most recent information
    - Is designed to be used on a desktop on either Chrome (newest) or IE (≥ 10) by a technical user
    - Its design is aligned with other CSL products (i.e. PatientFlow)

2.2. Out of Scope

- Mobile devices and tablets support
- Authorization of any form
  - The system is intended to be used on a closed network by a single user
- Unit information in the UX draft contains also information such as performance and MAE, these will be part of a next feature
  - UX draft assumes ordering units in client structure by MAE. Since MAE is not supported yet, it will be ordered alphabetically by unit name
- Storing units information in a database
  - Between the updates, all unit information will be stored in the memory
- Documentation such as IFS* - Interface specification

2.3. Assumptions, constraints and preconditions

It is assumed that the Forecast System can provide a list of all units information including readable unit name

2.4. Related issues

~~CSL-11302~~ - Readable name on forecast unit keys CLOSED

2.5. Documents to be updated

2.5.1. Maintained in CVS

Document number (DTS)	Name	Comment(s)

2.5.2. Maintained in Jira (R4J)

Requirement ID	Comment(s)

2.5.3. Maintained in Wiki-pages

Document number (DTS)	Link/name	Comment(s)

2.5.4. Draft Public Interface Description

3. Estimate

Initial estimate

Estimate for	Time
Clarification (Rfl+Rfc)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	50

Estimate after "Ready for Commitment" review

Estimate for	Time
Clarification (Rfl)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Implementation" review

Estimate for	Time
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	105

4. Feature Requirements

4.1. Requirements

- OS1: The solution should be designed for use by a Machine learning engineer or system admin in a hospital (technical minded person)
- OS2: The system should have an interactive front-end in form as a web application
- OS3: The solution system should be designed in such way, that it can easily be extended to monitor multiple forecast systems, which each contains multiple forecast pipelines at the same time.
- OS4: The system should provide an overview of all the monitored forecast systems

4.2. Acceptance Criteria

<<Table is suggested common accept criterias that should be considered for all features>>

#	Description
1	Web App displays units overview
2	Web App receives a regular update

5. Risks

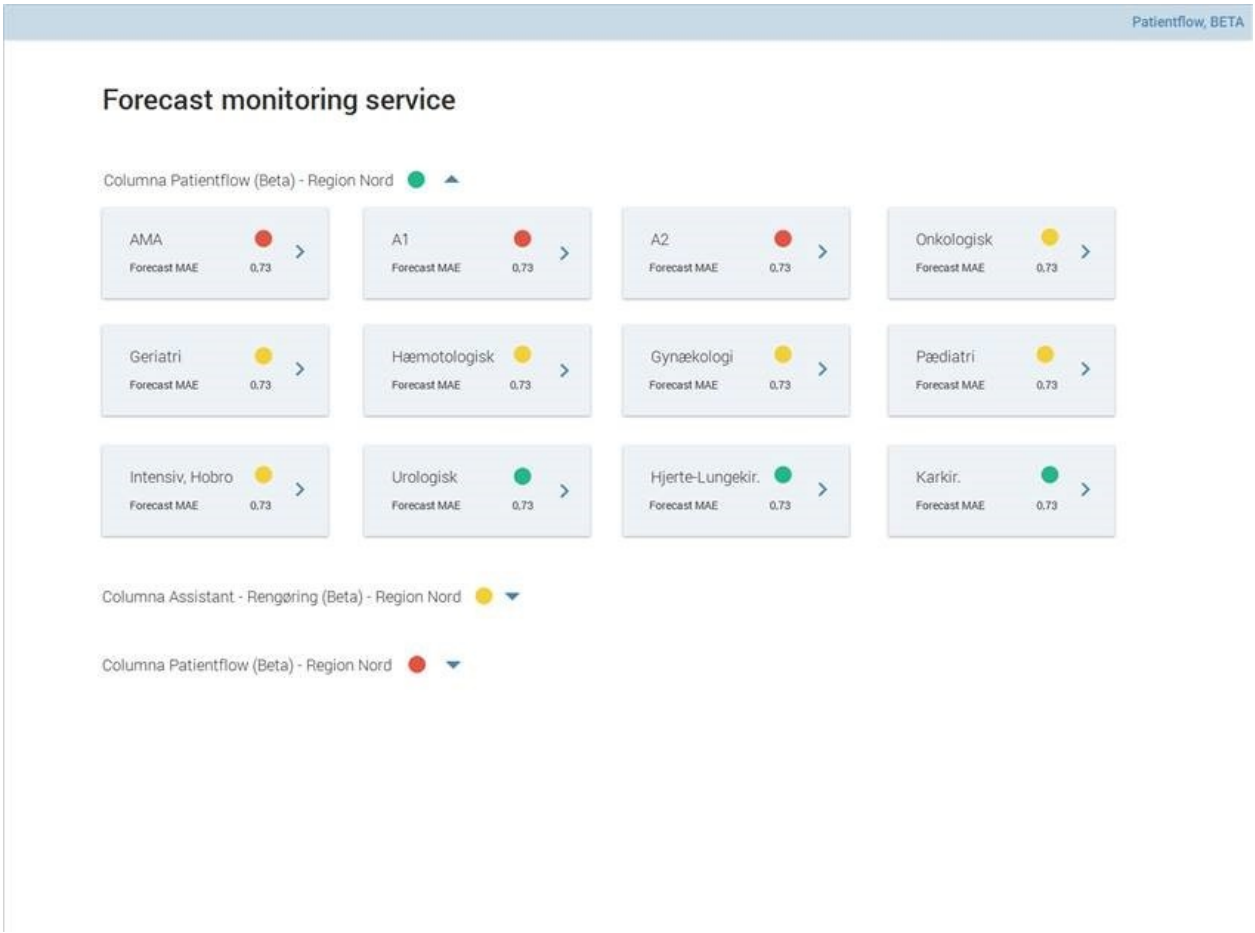
Description	Chance (%)	Cost (hours)	Mitigation plan	Cost of mitigation (hours)

6. Dependencies

7. Clarifications and actions

Clarification/action	Responsible	Result

8. User Interface Design



9. Technical Design

Web Service:

- store unit information in memory
- make instalations/regions configurable

Web App:

- make text items configurable (via translation file)
- save unit information in local storage (can be taken out and implemented later if too costly)

10. Deployment and operations

11. Test Approach

11.1. Test Design

No.	Test condition	Testtype (UI, service, integration, manual)	Test conducted with Story no. / Feature	Comments	Result
	The GUI supports the newest Chrome browser	Auto	Feature	Implicitly covered by Selenium tests	✅
	The system ensures an acceptable reconnection strategy to the Forecast System.	Integration	Feature		✅
	Simple sunshine scenario API call tests for the backend:	Integration	Feature	Success when none exception is thrown and an response is always received ForecastMonitor.Test.Integration.BusinessLogic.ForecastLogicTests: • GetInstallationsReturnsNoneNullResponse()	✅

	• Get Installations • Get Clients • Get units			• GetClientsWithNoneExistingInstallationIdReturnsEmptyObject() • GetUnitsWithNoneExistingInstallationAndClientIdReturnsEmptyObject()	
	The frontend ensures a responsible handling approach of the potential connection error to the backend	-	8	Not done, as a trustworthy code snippet from Angular's own site is used	✅
	The GUI layout matches the design draft	Manual	9		✅
	The GUI presents errorprone values will be handled in a user friendly way: • Units with null name field (printing out: Unit not found: unitKeyId) • Long unit names should be added ... and a tooltip (same like in PatientFlow)	Auto - Selenium	9		✅
	When server is unreacheable the web app will handle it in a user friendly way.	Auto - Selenium	9	Manual styling verification	✅
	The web app is implemented in such way, that the presented text can be configured via a translation file	Manual	10	Manual test to identify all the configured text labels with test translation file.	✅
	Periodical information update will automatically be presented on the web app	Integration + Manual	11	Publishing covered by integration test, client updates manually	✅

12. Stories and Other Activities

12.1. Story 1: Dummy REST Web Service

Implement REST Web Service (using .net core web api) with 3 dummy endpoints:

- /installations
  - returns object with list of installed systems (currently installation == Region)
  - format:

```
{
last_update: "CURRENT_TIME_UTC",
installations: [
{ name: "Columna PatientFlow (Beta) - Region Nord", id: 1 },
{ name: "Columna PatientFlow (Beta) - Region Midt", id: 2 }
]
}
```

- /clients?installationid=ID
  - returns object withlist of clients in given installation
  - ID = 1:
    - format:

```
{
```

last_update: "CURRENT_TIME_UTC",

clients: [

{ name: "Cleaning Capacity", id: 1 },

{ name: "Patient Capacity", id: 2 }

]

}

- ID = 2:
  - format: same as 1 but one client only { name: "Bed Capacity", id: 1 }
- /units?installationid=ID&clientid=ID
  - returns object with list of units for given installation and client
  - add dummy data for all combinations specified above in following format:
    - format:

{

last_update: "CURRENT_TIME_UTC",

units: [

{ name: "Unit 1", id: 1 },

{ name: "Unit 2", id: 2 },

{ name: "AMA", id: 3 },

{ name: "One very long unit name", id: 4 }

]

}

12.1.1. Completion criteria

- Dummy data are coming from DataService (not a controller)
- All 3 endpoints are alive
- ~~Serialization done via MessagePack~~

12.1.2. Test approach

As the use of the API is internal, and it is the assumption that the calls will be done by developer who knows the internal of the API, it is hence decided that API call with the wrong format or params will not bring noticeable value.

- Simple sunshine scenario API call tests
- NUnit tests for development purpose

12.1.3. Estimate

- 50SP

12.2. Story 2: API Help Page

Create new endpoint using Swagger UI

- url: /help (by default /swagger - perhaps can't be changed?)

12.2.1. Completion critera

- hitting /help displays Swagger UI API description

12.2.2. Test approach

- Content review from the primary user of this page. (The front-end developer)

12.2.3. Estimate

- 10SP

12.3. Story 3: Installations endpoint returns live data

Installations are configured via appsettings.json using template:

"Installations" : [

{

"name": "Test1 Forecast System",

"url": "http://dabai-test1.systematicgroup.local:6060"

}

]

installationid is $index + 1

test1 url is used for debug config, leave release empty

12.3.1. Completion criteria

- DataService returns names based on data specified in appsettings.json
- appsettings.json supports transformation (platform independent)
- ~~forecast system (FS) implemented as internal service (connection logic to FS is isolated from the internal DataService implementation)~~ - Pushed to story 4

12.3.2. Test approach

12.3.3. Estimate

- 20SP

12.4. Story 4: Clients endpoint returns live data

For each installation, clients enpoint calls to /clients/all and uses the result as follows:

- "client_id" => id
- "clien_key" => name

12.4.1. Completion criteria

- DataService returns data based on data from forecast system
- ~~400 if installation id does not exist (or 404?)~~ - not needed as this will be handled in story #6
- 503 if FS is unavailable (or just the code received from FS)

12.4.2. Test approach

- NUnit

12.4.3. Estimate

- 5SP

12.5. Story 5: Units endpoint returns live data

units enpoint uses unit_keys/all to get all unit keys, filters them based on cliend id and uses the result as follows:

- "name" => name
- "unit_key_id" => id

12.5.1. Completion criteria

- DataService returns data based on data from forecast system
- 400 if installation id or client id does not exist (or 404?)
- 503 if FS is unavailable (or just the code received from FS)

12.5.2. Test approach

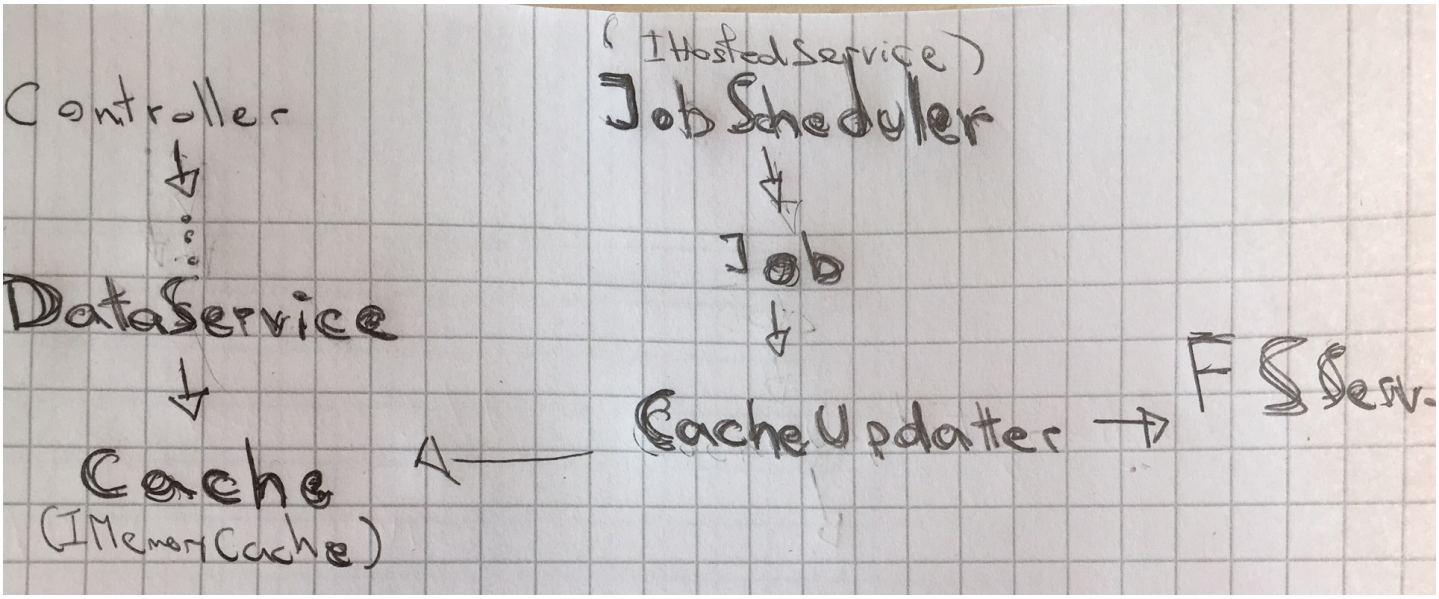
- NUnit

12.5.3. Estimate

- 5SP

12.6. Story 6: FS results caching and job scheduling

- make FS calls every x minutes (via appsettings.json)
- scheduling implemented via IHostedService
- store results between calls in IMemoryCache



12.6.1. Completion Criteria

- On application start, cache gets populated by data from FS, then is updated periodically
- Updates are logged
- In case of FS unavailability, job retries to execute after 1, 3, 5 minutes (similar to TCP)
- DataService calls do always hit cache

12.6.2. Test approach

- NUnit

12.6.3. Estimate

- 50SP

12.7. Story 7: Building web app's layout based on the GUI draft (with dummy data)

- Implement header and footer to match the Patient Flow color scheme
- Design the main page (dashboard) to match the GUI draft received from the UXer
- Display dummy data for the Installations and regions to visualize the design

12.7.1. Completion criteria

- The layout of the three main components match the GUI draft
- There are no overlaps of elements on the page

12.7.2. Test approach

No specific story test coverage will be done, the content will be tested with story 9.

12.7.3. Estimate

- 10SP

12.8. Story 8: Implement a service to retrieve data from the web service

Implement a service with three main functions

- getInstallations()
  - accepts no parameters
  - sends an HTTP GET request to /installations
  - returns object with list of installed systems
    - format
    - ```
      {
          last_update: "CURRENT_TIME_UTC",
          installations: [{
              name: "Columna PatientFlow (Beta) - Region Nord",
              id: 1
          }, {
              name: "Columna PatientFlow (Beta) - Region Midt",
              id: 2
          }]
      }
      ```
- getClients(id)
  - accepts one integer parameter (id)
  - sends an HTTP GET request to /clients?installationId=id
      ```

- returns an object with list of clients in given instalation
  - format

```
{
    last_update: "CURRENT_TIME_UTC",
    clients: [{
        name: "Cleaning Capacity",
        id: 1
    }, {
        name: "Patient Capacity",
        id: 2
    }]
}
```

- getUnits(installation_id, client_id)
  - accepts two integer parameters (installation_id, client_id)
  - sends and HTTP GET request to /units/?installationId=installation_id&clientId=client_id
  - returns an object with list units for a given client in a given instalation
    - format

```
{
    last_update: "CURRENT_TIME_UTC",
    units: [{
        name: "Unit 1",
        id: 1
    }, {
        name: "Unit 2",
        id: 2
    }, {
        name: "AMA",
        id: 3
    }, {
        name: "One very long unit name",
        id: 4
    }]
}
```

12.8.1. Completion criteria

- Each function successfully hits the endpoint and retrieve data in the specified format
- Errors that results from an invalid parameters or server unavailability should be handled

12.8.2. Test approach

Manual GUI test will be done with story 9.

Unit test for server call error handling not prioritised as trustworthy library is used to handle this part of Business Logic

12.8.3. Estimate

- 50SP

12.9. Story 9: Display the data retrieved from the web service on the web app

Replace the dummy data used in Story 7 with the data retrieved from the web service

12.9.1. Completion criteria

- All installations, clients and units are displayed in the correct hierarchy as received from the web service
- Long names are truncated and a tool tip is shown

12.9.2. Test approach

Manual user experience test to:

- Prove the GUI layout matches the design draft (story 7 content)
- Prove when server is unreacheable the web app will handle it in a user friendly way (partial story 8 content)
- Prove data presentation of errorprone values will be handled in a user friendly way (partial story 8 content)
  - Units without name (printing out: Unit not found: unitKeyId)

12.9.3. Estimate

- 5SP

12.10. Story 10: Create a translation service to fetch data from a translation file and display it on the web app

Fetch all text data from a translation file through a service

12.10.1. Completion criteria

- There are no hard coded text values on the web app
- All text values are fetched from a translation file provided beforehand

12.10.2. Test approach

Manual test to identify possible uncovered string values with test translation file.

12.10.3. Estimate

- 5SP

12.11. Story 11: Establish a socket connection between the web service and the web app

Establish a socket connection between the back-end and front-end using SignlaR and push regular data about unit data changes.

Backend:

- publish new units to all clients at:
  - route: /hub/units
  - topic: "UnitsUpdate"

12.11.1. Completion criteria

- The front-end successfully subscribes on the back-end socket when on the dashboard and listens for messages
- Back-end sends regular messages containing the new unit data when retrieved from FS.

12.11.2. Test approach

12.11.3. Estimate

13. Background Information

Reference (e.g. DTS, link,...)	Alias	Title

FDD0001 Display unit information

Created by Ophelia Marott Zhang, last modified on 2018-11-23

NOTE: To use this template, draw a number for the new FDD in DTS. Save a copy of this page with the title "FDD<DTS no.> <Feature name>" in the Wiki-space where your project keeps FDDs. Then read the Template user's guide below for more information.

Overview

Feature Lead

@Tomas Izo

State	Responsible	Participants in Review
RFC	@Ophelia Marott Zhang	@Adam René Gregersen @Tomas Izo @Georgi Stefanov
RFI	@Tomas Izo	@Ophelia Marott Zhang @Georgi Stefanov @Adam René Gregersen

Jira

~~CSL-11564~~ - FDD0001 Display unit information CLOSED

Content

1. Feature Statement

After this feature, the user will have overview of all the monitored forecast pipline/model performance in form as MAE scores (calculated between historical forecast and the actual historical value).

2. Scope and Limitations

2.1. Scope

FS = the Forecast System

1. Performance indication by comparing current unit's model MAE with historical models MAE using stdev:
   a. Get data from FS (per unit key):
      i. get the latest actual timeserie
      ii. get the model with prediction at that time (latest evaluable model)
      iii. get a list of all the predictions ever made by that model
      iv. Per unit_key, get a list of all the actual historical value from FS.
      v. Get also predictions up to two weeks before the first prediction of latest model
      vi. Get timeseries that spans through whole prediction period (interval [two weeks before first latest

        ii. The implementation for data processing should be implemented so it can be extended to calculate other metrics than MAE in the future.
    b. Calculate Std of current model's MAE with historical AE.
    c. Compare current unit model performance to historical performance, where:
        i. Red - indicates MAE≥3std
        ii. Yellow - indicates 1.5std≤MAE<3std
        iii. Green - indicates MAE<1.5std
2. GUI:
    a. Presentation of MAE scores per unit on the front end Web app
    b. Sorting:
        i. Sort presented unit according to first their performance indication color, then MAE scores
        ii. Sort presented client according to first their performance indication color, then their worst performing unit's MAE scores
        iii. Sort presented region according to first their performance indication color, then their worst performing client's MAE scores
    c. Color performance indication:
        i. Each unit has a color indication for its current prediction performance
        ii. Clients has the same color indication as its worst performing unit
        iii. Regions has the same color indication as its worst performing client
    d. Unfold any client with red performance indication automatically
    e. The frontend automatically tries to reconnect to the backend in case of connection error with configurable policy.
3. Logging
    a. Logging for incoming (front-end) and outgoing (forecast system) service call and response
    b. Logging for internal data processing (MAE calculation) if needed

2.2. Out of Scope

- Performance optimization for data presentation

2.3. Assumptions, constraints and preconditions

Endpoints exposed by the Forecast System

In order to complete this feature, the forecast system must provide endpoints to access the data needed for performance calculation:

- To get all the predictions made by one model in given interval:
    - http://dabai-test1.systematicgroup.local:6060/predictions?from_dt=2018-09-10T16:00:00.0&to_dt=2018-09-24T16:00:00.0&model_id=19618
- To get all the predictions for one unit in given interval for the unit:
    - http://dabai-test1.systematicgroup.local:6060/predictions?from_dt=2018-10-01T00:00:00.0&to_dt=2918-10-15T00:00:00.0&unit_key=0452B832-85BC-4457-84BF-738B509DDE45_PatientsAtDepartment
- To get the most recent historical label (value) for the unit:
    - http://dabai-test1:6060/timeseries/?client_key=PatientCapacity&unit_key=0452B832-85BC-4457-84BF-738B509DDE45_PatientsAtDepartment&most_recent=1
- To get prediction at exact time:
    - http://dabai-test1.systematicgroup.local:6060/predictions?from_dt=2018-09-24T16:00:00.0&to_dt=2018-09-24T16:00:00.0&unit_key=0452B832-85BC-4457-84BF-738B509DDE45_PatientsAtDepartment

Std (standard deviation) definition

In this feature the standard deviation is noted as std, and is defined by: (s in the picture)

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}.$$

- x_i = observed values
- x dash = mean value of these observations

In this feature the mean absolute error is denoted as MAE, and is defined by the following statement, where:

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

- Y = actual historical value
- X = predicted value
- n = number of observations

2.4. Related issues

Must be deployed to dabai-test1

- [x] ~~CSL-12183~~ - Add filtering option to the timeseries endpoint CLOSED

2.5. Documents to be updated

2.5.1. Maintained in CVS

Document number (DTS)	Name	Comment(s)

2.5.2. Maintained in Jira (R4J)

Requirement ID	Comment(s)

2.5.3. Maintained in Wiki-pages

Document number (DTS)	Link/name	Comment(s)

2.5.4. Draft Public Interface Description

3. Estimate

Initial estimate

Estimate for	Time

Estimate for	Time
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Commitment" review

Estimate for	Time
Clarification (RfI)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Implementation" review

Estimate for	Time
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

4. Feature Requirements

4.1. Requirements

- OS5: For each monitored system, there should be an overview of all the monitored forecast pipelines
- PP1: The solution system should be able to display the Prediction Performance (in form as different evaluation metrics, only MAE score is guaranteed supported in this project) of the historical forecast compared to the historical result in intervals that is configurable, but two weeks as the default backwards.
- PP4: The solution system should be able to display performance information in a sorted manner, so the presentation of Region, Client and Unit are ordered by first the severity of the performance indication, then MAE, both in descending

- a. Each unit has a color indication for its current prediction performance compared to standard deviation of historical MAE distribution, where:
    - i. Red - indicates MAE≥3std
      ii. Yellow - indicates 1.5std≤MAE<3std
      iii. Green - indicates MAE<1.5std
- b. Clients has the same color indication as its worst performing unit
- c. Regions has the same color indication as its worst performing client

- PP6: The solution system will automatically unfold any client with red performance indication, while the rest remains folded.

4.2. Acceptance Criteria

<<Table is suggested common accept criterias that should be considered for all features>>

#	Description
1	OS5: As a user of the system I have overview for all the units.
2	PP1: As a user of the system I have overview for each unit's active model's performance in form as MAE scores
3	PP4: As a user of the system I get Region, Client and Unit presented in a sorted manner by first the severity of the performance indication, then MAE, both in descending order.
4	PP5: As a user of the system I can see current prediction performance indication in color code.
5	PP5: As a user of the system I can see red performance indication for a unit when MAE≥3std
6	PP5: As a user of the system I can see yellow performance indication for a unit when 1.5std≤MAE<3std
7	PP5: As a user of the system I can see green performance indication for a unit when MAE<1.5std
8	PP5: As a user of the system I can see the client's worst performing unit's color indication on the client
9	PP5: As a user of the system I can see the region's worst performing client's color indication on the region
10	PP6: As a user of the system I can expect any client with red performance indication to be unfolded automatically on the GUI.

5. Risks

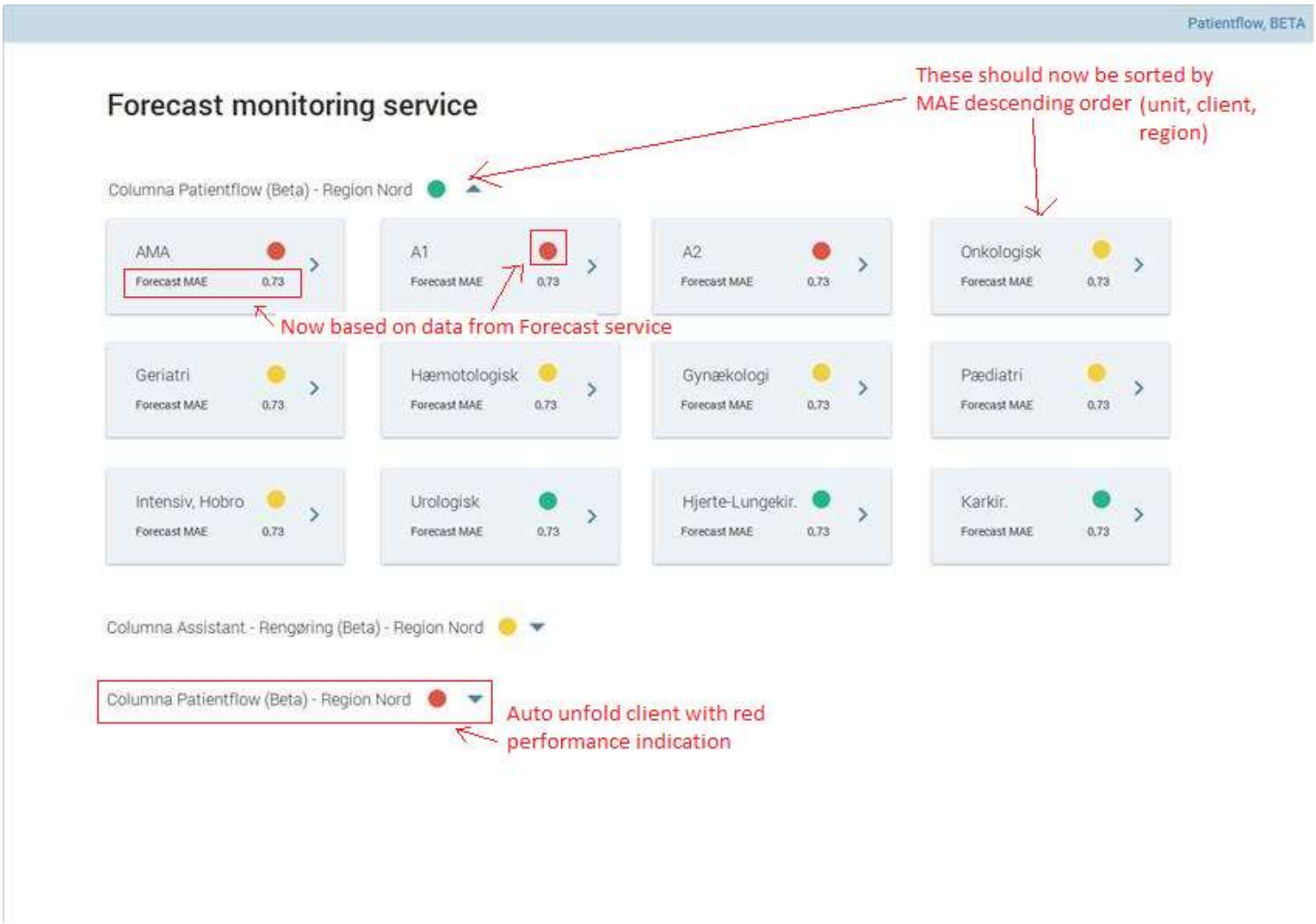
Description	Chance (%)	Cost (hours)	Mitigation plan	Cost of mitigation (hours)

6. Dependencies

7. Clarifications and actions

Clarification/action	Responsible	Result
Make sure communication to Forecast System, so they offer all of the endpoints needed for implementation of this feature	@ Ophelia Marott Zhang	Done, see section 2.3
Clarify if MAE should be calculated model specific or not.	@ Ophelia Marott Zhang	MAE should be model specific.

8. User Interface Design



9. Technical Design

In the very high level. The backend implementation is "isolated" by the data cache:

- The request from the front end is always answered with the data currently available in the data cache
- The data cache is updated by the background job every specific time interval

Backend requires changes:

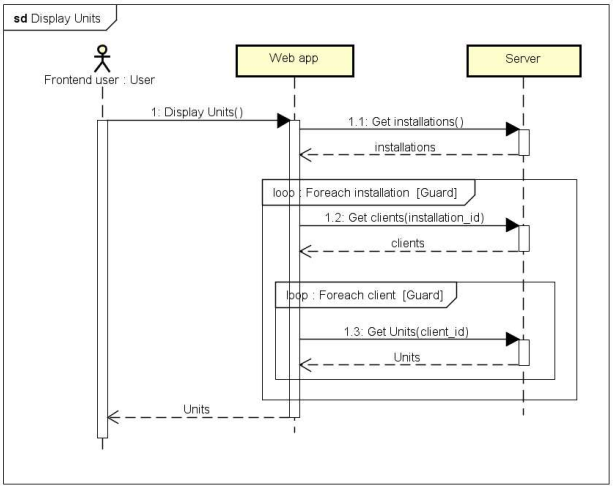
2. There has to added a new DomainLogic package for handling the performance calculation logic
3. The background job for updating data cache has to be updated so it supports the change in point 1.
4. The background job for updating data cache has to be updated so it calls the logic implemented in point 2 after data retrieval.
5. New/update data models in DataAccessLogic.DataAccessObjects package to support internal data processing.
6. Update data model Unit in BusinessLogic.DataTranferObjects package
7. New/update data mappers
8. Update the data cache implementation, so it can contain new objects from point 5.

Frontend requires changes:

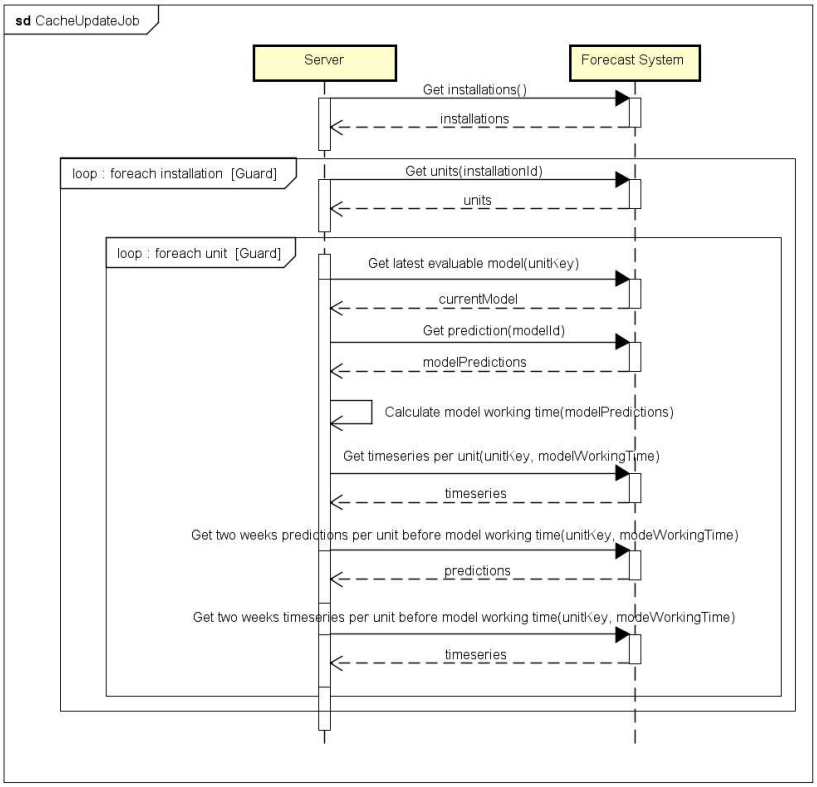
1. Change to accept unit in the new DTO format
2. Sorting components of all levels according to performance indication and MAE performance
3. Auto expanding all components with red performance indication
4. Use live data for MAE and performance indication

9.1. Sequence diagram

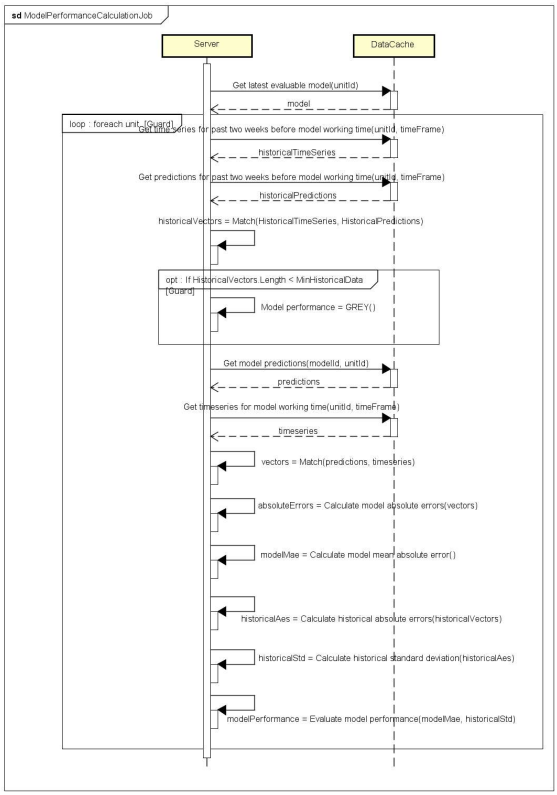
9.1.1. Display Units (including performance data)



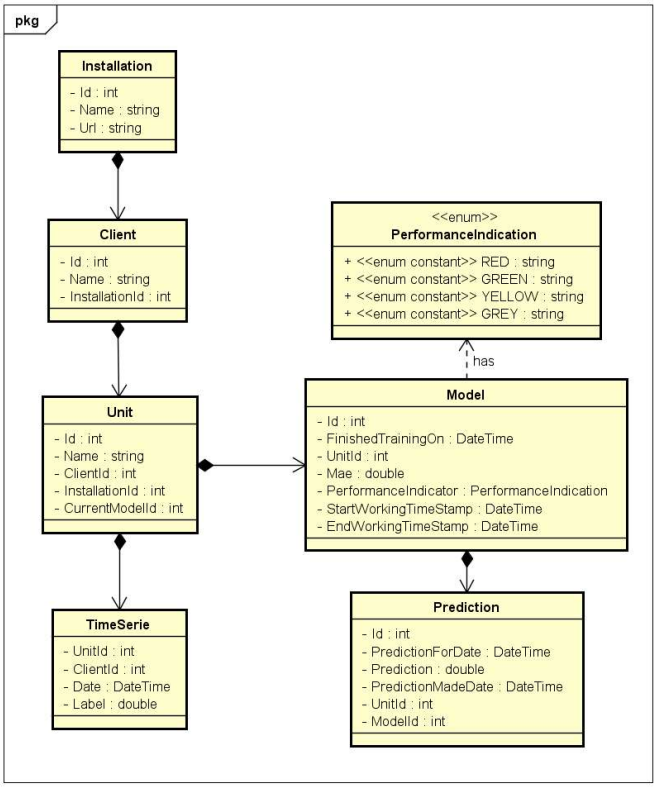
9.1.2. Update Cache Job



9.1.3. Calculation Job



9.2. Data access object class diagram



10. Deployment and operations

11. Test Approach

11.1. Test Design

No.	Test condition	Test type (UI, service, integration, manual)	Test conducts with Story no. / Feature	Comments	Result
OS5-1	For each monitored system, there should be an overview of all the monitored forecast pipelines	Manual	Feature	TSP	✅
PP1-1	The solution system should be able to display the Prediction Performance as MAE score.	Manual	Feature	TSP	✅
PP1-2	The MAE score is calculated from comparing the historical forecast to the historical result in intervals that is configurable, but two weeks as the default backwards.	Unit	2	Backend, set to 28 days due to lack of data on test server	✅

	Region, Client and Unit are ordered by first the severity of the performance indication, then MAE, both in descending order.				
PP5-1	The system indicates current prediction performance in color code, according to: • i. Red - indicates MAE≥3std ii. Yellow - indicates 1.5std≤MAE<3std iii. Green - indicates MAE<1.5std	Unit	2	Test for colorcode "calculation" logic at backend.	✅
PP5-2	The system displays client performance indication as its worst performing unit.	GUI	4	Front end	✅
PP5-3	The system displays region performance indication as its worst performing client.	GUI	4	Front end	✅
PP6-1	The solution system will automatically unfold any client with red performance indication.	GUI	4	Front end	✅

12. Stories and Other Activities

12.1. Story 1: Extend current data retrieval logic

~~CSL-12176~~ - Story 1: Extend current data retrieval logic CLOSED

Add methods to ForecastSystemClient:

- to get all active models
- to get predictions for unit
- to get historical timeseries
- to get historical predictions

Use new methods in CacheUpdateJob

12.1.1. Completion criteria

Be able to fetch from all the stated endpoints above and map them to DTOs.

CacheUpdateJob stores results

12.1.2. Test approach

12.1.3. Estimate

12.2. Story 2: Performance calculation logic

~~CSL-12177~~ - Story 2: Performance calculation logic CLOSED

12.2.1. Completion criteria

12.2.2. Test approach

12.2.3. Estimate

12.3. Story 3: Performance calculation job and job mediator

~~CSL-12178~~ - Story 3: Performance calculation job and job mediator CLOSED

Create a PerformanceCalculationJob that is triggered after CacheUpdateJob completes.

Once the PerformanceCalculationJob completes, call a UnitPublishingJob.

Do not call those jobs directly from their previous job, use an event handler instead (MediatR)

12.3.1. Completion criteria

Having a notification handler which can run ad-hoc jobs upon request

12.3.2. Test approach

12.3.3. Estimate

12.4. Story 4: Sort presented components according to MAE score

~~CSL-12180~~ - Story 4: Sort presented components according to MAE score CLOSED

Backend:

Update data model Unit in BusinessLogic.DataTranferObjects package to:

```
unit: {
    id : int,
    name : string,
    client_id : int,
    installation_id : int,
    mae : double,
    performance_indication : {Red, Green, Yellow}
}
```

Update existing unit data mapper for DAO and Frontend DTO.

Frontend:

Accept unit object in the new DTO format

Assign client and region performance indicator according to worst performing unit.

Sorting components of all levels according to MAE performance

Auto expanding all components with red performance indication

Use live data for MAE and performance indication

12.4.2. Test approach

12.4.3. Estimate

12.5. Story 5: Automatic reconnection

~~CSL-12181~~ - Story 5: Automatic reconnection CLOSED

The frontend automatically tries to reconnect to the backend in case of connection error with configurable policy.

12.5.1. Completion criteria

The frontend now automatically tries to reconnect to the backend according to the configured reconnection policy.

12.5.2. Test approach

12.5.3. Estimate

12.6. Other activity 1: <Name>

12.6.1. Completion criteria

12.6.2. Estimate

13. Background Information

Reference (e.g. DTS, link,...)	Alias	Title

Pages / ... / Completed 7 Jira links

FDD0002 - Display unit prediction performance evaluation plot

Created by Georgi Stefanov, last modified on 2018-11-26

NOTE: To use this template, draw a number for the new FDD in DTS. Save a copy of this page with the title "FDD<DTS no.> <Feature name>" in the Wiki-space where your project keeps FDDs. Then read the Template user's guide below for more information.

Overview

Feature Lead

@ Georgi Stefanov

State	Responsible	Participants in Review
RFC	@ Georgi Stefanov	@ Adam René Gregersen @ Tomas Izo @ Ophelia Marott Zhang
RFI	@ Georgi Stefanov @ Tomas Izo	@ Adam René Gregersen @ Ophelia Marott Zhang

Jira

Content

Click here to expand...

1. Feature Statement

After this feature the user will able to see an evaluation plot for each unit, comparing the forecast result and the actual result in intervals that is configurable. but two weeks as the default backwards as two different series in one graph.

2. Scope and Limitations

2.1. Scope

FS = the Forecast System

- Gather and process data from the FS:
  - Get data set of all performance predictions and actual values for the requested interval and unit
  - Apply any data processing to the data set, required for better visualization on the web app
- Back-end:
  - Provide endpoints for retrieving performance data for a specific unit in a specific interval
  - API response object to a flat structure
- Front-end:
  - Gather required performance data from the back-end
  - Apply any data manipulation needed to display it in the used library
  - Visualize the evaluation plot in a single graph with multiple series using Line chart in the ngx-charts library
  - Follow CSL color theme for the graph
  - Adjust the interfaces to work with the changed response objects from the back-end

2.2. Out of Scope

- Performance optimization for data retrieval and presentation

2.3. Assumptions, constraints and preconditions

In order to complete this feature the FS must provide endpoints to access the data needed for visualizing the evaluation plot:

- To get prediction performance data for a specific unit in a certain interval:
- To get actual performance data for a specific unit in a certain interval:

2.4. Related issues

2.5. Documents to be updated

2.5.1. Maintained in CVS

Document number (DTS)	Name	Comment(s)

2.5.2. Maintained in Jira (R4J)

Requirement ID	Comment(s)

2.5.3. Maintained in Wiki-pages

Document number (DTS)	Link/name	Comment(s)

2.5.4. Draft Public Interface Description

3. Estimate

Initial estimate

Estimate for	Time
Clarification (RfI+Rfc)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	

Estimate for	Time
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Commitment" review

Estimate for	Time
Clarification (RfI)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Implementation" review

Estimate for	Time
Development + Inspection (Stories)	116
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	116SP

4. Feature Requirements

4.1. Requirements

- PP2: The system should be able to display an evaluation plot, comparing the forecast result and the actual result in intervals that is configurable, but two weeks as the default backwards as two different series in one graph.

4.2. Acceptance Criteria

<<Table is suggested common accept criterias that should be considered for all features>>

#	Description
1	PP2: As a user I can see an evaluation plot for a selected unit for the last 2 weeks by default
2	PP2: As a user I can configure the interval for which I want to see the evaluation plot for the selected unit
3	The charts are styled to follow the CSL theme

5. Risks

Description	Chance (%)	Cost (hours)	Mitigation plan	Cost of mitigation (hours)

6. Dependencies

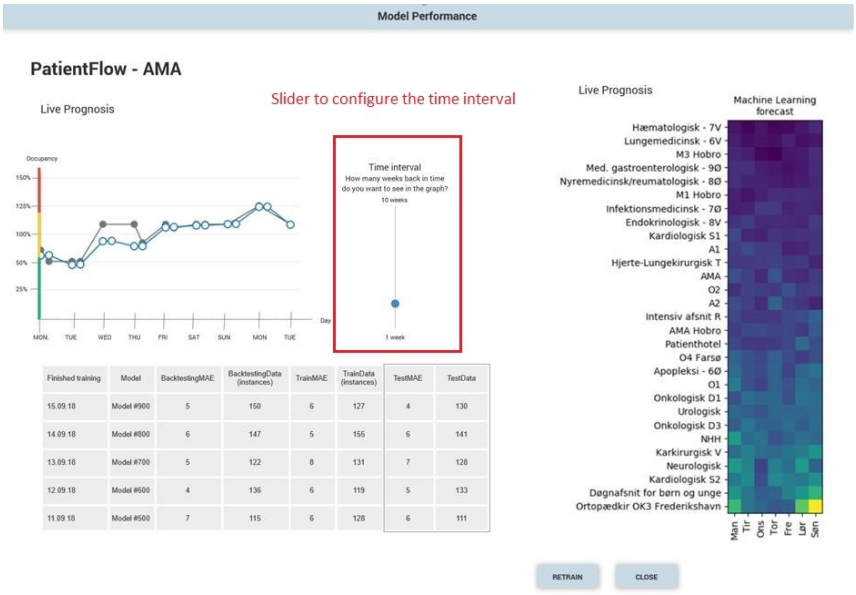
7. Clarifications and actions

Clarification/action	Responsible	Result

Clarification/action	Responsible	Result
Does the performance data from the FS requires any data processing before passed to the web app	@Ophelia Marott Zhang	Yes, for the precentage and the weekday names.
What type and structure of data is needed for the ngx-charts visualization library	@Georgi Stefanov	Each entry to the chart needs to have a name and a list of series containing name (x axis position) and value (y axis position)
How should the configuration of the time interval should be done? Via configuration file or UI element?	@Georgi Stefanov	Via slider in the UI.

8. User Interface Design

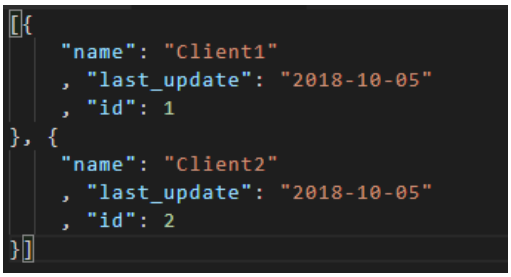
When a unit card is selected from the list, the user should be presented with this view



9. Technical Design

Restructuring of the API response object is required. Instead of having a last_update timestamp for the whole response we have it for each element in the response because they can get updated separately. After removing the "global" timestamp we can move to a flat structure:

```
{
    "clients": [{
        "name": "Client1"
        , "last_update": "2018-10-05"
        , "id": 1
    }, {
        "name": "Client2"
        , "last_update": "2018-10-05"
        , "id": 2
    }]
    , "last_update": "2018-10-05"
}
```



```
[{
    "name": "Client1"
    , "last_update": "2018-10-05"
    , "id": 1
}, {
    "name": "Client2"
    , "last_update": "2018-10-05"
    , "id": 2
}]
```

Following the same pattern the Installations and Units responses should be updated too.

10. Deployment and operations

11. Test Approach

~~CSL-12349~~ - FDD0002 - Test Design CLOSED

11.1. Test Design

No.	Test condition	Testtype (UI, service, integration, manual)	Test conducts with Story no. / Feature	Comments	Result
PP2-1	As a user I can see an evaluation plot for a selected unit for the last 2 weeks by default	Manual	3	TSP	✅
PP2-2	As a user I can configure the interval for which I want to see the evaluation plot for the selected unit	Manual	4	TSP	✅
PP2-3	The graph should change scale according to the data to be presented.	GUI	4	Auto scaling is handled by the library	✅
	Test with error-prone data set:		3	Frontend	✅
	▪ Empty data	Selenium	3	Frontend	✅

	▪ Few data points:
	  ▪ Only data in beginning of periode
	  ▪ Only data in end of periode
	  ▪ Data spreaded over periode
			3	Rendering the graph is handled by external library and testing it is not adding any value	✅

	▪ Uneven prediction and actual data		3	Same as above	✅

12. Stories and Other Activities

12.1. Story 1: Retrieve performance data

~~CSL-12289~~ - FDD0002 - Story 1: Retrieve performance data CLOSED

Retrieve performance data from the back-end service by specifying unit_id, from and to (dates for the time interval). Expected response object should look like this or similar:

```
{
  unit_name: 'Test Unit 1',
  ward_capacity: 1500, // The capacity of the ward so a % can be calculated
  name: 'Prediction',
  series: [
    {
      name: 'Mon', // Day of the week
      value: 500 // raw occupancy number
    },
    {
      name: 'Tue',
      value: 570
    }
  ]
}, {
  name: 'Actual',
  series: [
    {
      name: 'Mon',
      value: 505
    },
    {
      name: 'Tue',
      value: 572
    }
  ]
}
```

12.1.1. Completion criteria

When a unit is selected performance data for a period of 2 weeks by default should be retrieved.

12.1.2. Test approach

No.	Test condition	Testtype (UI, service, integration, manual)	Comments	Result
PP2-1	As a user I can see an evaluation plot for a selected unit for the last 2 weeks by default	service	unit	✅

12.1.3. Estimate

20SP

12.2. Story 2: Front-end flatten interfaces

~~CSL-12290~~ - FDD0002 - Story 2: Front-end flatten interfaces CLOSED

Change the interfaces on the front end to accept the flat response structure

```
[{
    "name": "Client1"
    , "last_update": "2018-10-05"
    , "id": 1
}, {
    "name": "Client2"
    , "last_update": "2018-10-05"
    , "id": 2
}]
```

12.2.1. Completion criteria

When data is received by the back-end it will be in the above format and the front-end will handle it without any errors

12.2.2. Test approach

12.2.3. Estimate

5SP

12.3. Story 3: Visualize the performance data

~~CSL-12291~~ - FDD0002 - Story 3: Visualize the performance data CLOSED

Design the page and graph following the UI draft and display the performance data on a line graph

12.3.1. Completion criteria

When a unit is selected a page is displayed containing the unit name on the top and a performance (line) graph showing the performance data for a period of 2 weeks by default.

The graph and the page follows the CSL color theme

12.3.2. Test approach

No.	Test condition	Testtype (UI, service, integration, manual)	Comments	Result
PP2-1	As a user I can see an evaluation plot for a selected unit for the last 2 weeks by default	UI	Frontend	✅
	Test with error-prone data set:			
	• Empty data	selenium	Description: "Empty data"	✅

	• Few data points:
	  • Only data in beginning of periode
	  • Only data in end of periode
	  • Data spreaded over periode
			Rendering the graph is handled by external library and testing it is not adding any value	✅

	• Uneven prediction and actual data		Same as above	✅

12.3.3. Estimate

60SP

12.4. Story 4: Configure time interval

~~CSL-12292~~ - FDD0002 - Story 4: Configure time interval CLOSED

Implement a slider and allow the user to configure the time interval for the retrieved performance data

12.4.1. Completion criteria

On the unit performance page on the right side of the performance graph there is a slider showing the current time interval. Upon change the performance draft is redrawn with the new data.

12.4.2. Test approach

No.	Test condition	Testtype (UI, service, integration, manual)	Comments	Result
PP2-2	As a user I can configure the interval for which I want to see the evaluation plot for the selected unit	manual	4	✅
	The graph should change scale according to the data to be presented	GUI	Auto scaling is handled by the library (ngx-charts)	✅

12.4.3. Estimate

20SP

12.5. Story 5: Flatten REST service json responds

~~CSL-12347~~ - FDD0002 - Story 5: Flatten REST service json responds CLOSED

just like in story 2

12.5.1. Completion criteria

12.5.2. Estimate

1SP

12.6. Story 6: Implement chart retrieval logic and expose by the service

~~CSL-12348~~ - FDD0002 - Story 6: Implement chart retrieval logic and expose by the service CLOSED

the endpoint will be: unit/plot?installationId=<int>&unitId=<int>&weeksAgo=<int>:

- weeks_ago is optional and if not specified, endpoint returns data points for last 2 weeks

12.6.1. Completion criteria

- user can get data points with predictions and timeseries by specific date range
- default date range is specified in appsettings.json
- return object matches figure from story 1

12.6.2. Estimate

10SP

13. Background Information

Reference (e.g. DTS, link,...)	Alias	Title

FDD0004 - Trigger model retrain for a certain unit

Created by Georgi Stefanov, last modified on 2018-12-07

NOTE: To use this template, draw a number for the new FDD in DTS. Save a copy of this page with the title "FDD<DTS no.> <Feature name>" in the Wiki-space where your project keeps FDDs. Then read the Template user's guide below for more information.

Overview

Feature Lead

@Georgi Stefanov

State	Responsible	Participants in Review
RFC	@Georgi Stefanov	@Tomas Izo @Ophelia Marott Zhang @Adam René Gregersen
RFI	@Georgi Stefanov	@Tomas Izo @Ophelia Marott Zhang @Adam René Gregersen

Jira

~~CSL-12832~~ - Trigger model retrain for a certain unit CLOSED

Content

Pages / ... / Completed 3 Jira links

1. Feature Statement

After this feature the user will be able to trigger a model retrain for a certain unit with one button click

2. Scope and Limitations

2.1. Scope

FS = Forecast System

- Back-end:
  - Create and expose a new endpoint for the front-end to trigger model retrain on FS
- Front-end:
  - Add a button in the unit performance page for triggering a model retrain
  - Create a function that sends a request to the newly created back-end endpoint for retraining a model

2.2. Out of Scope

- Following the model training progress
- Actively notifying the user once the training is done

2.3. Assumptions, constraints and preconditions

2.4. Related issues

2.5. Documents to be updated

2.5.1. Maintained in CVS

Document number (DTS)	Name	Comment(s)

2.5.2. Maintained in Jira (R4J)

Requirement ID	Comment(s)

Pages / ... / Completed 3 Jira links

2.5.3. Maintained in Wiki-pages

Document number (DTS)	Link/name	Comment(s)

2.5.4. Draft Public Interface Description

3. Estimate

Initial estimate

Estimate for	Time
Clarification (RfI+Rfc)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Commitment" review

Estimate for	Time
Clarification (RfI)	
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	
Total	

Estimate after "Ready for Implementation" review

Estimate for	Time
Development + Inspection (Stories)	
Feature Done (Test,Bugfix,Write documentation)	
Operations (Deployment and delivery)	
Risk uncertainty	

4. Feature Requirements

4.1. Requirements

- OS7: The system should be able to provide a way (possibly via button click) to trigger a new training process for one specific pipeline

4.2. Acceptance Criteria

<<Table is suggested common accept criterias that should be considered for all features>>

#	Description
1	OS7: As a user I can trigger a model retrain using a button located on the model performance page

5. Risks

Description	Chance (%)	Cost (hours)	Mitigation plan	Cost of mitigation (hours)

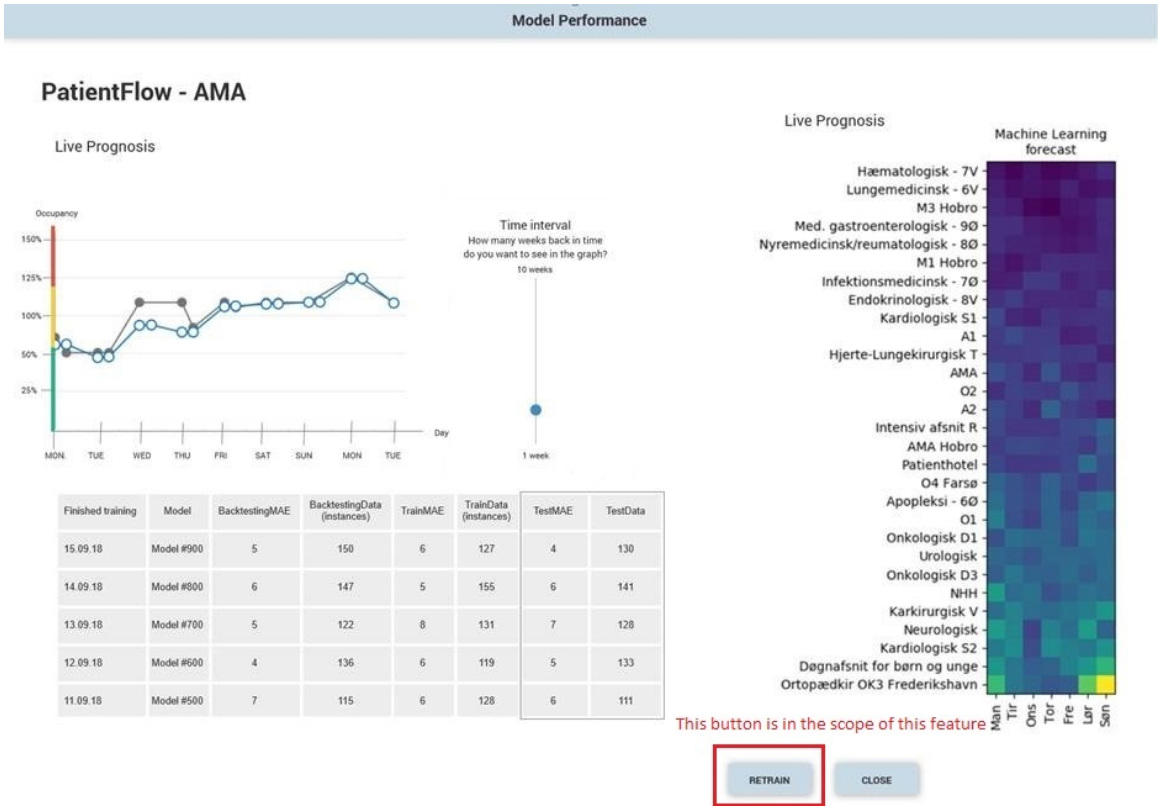
6. Dependencies

7. Clarifications and actions

Clarification/action	Responsible	Result
The retrain should be only for the currently active model of the unit (last model)	@ Georgi Stefanov	Retrains only the last model of the pipeline
How long does it take for a model to complete training?	@ Georgi Stefanov	Retraining doesn't have a fixed time. It can be 1 minute or 1 hour.
Does the FS provide a feedback when the training is complete?	@ Georgi Stefanov	The FS doesn't provide a push feedback that the training is complete, but it does change the 'last_update' timestamp.
How should the Forecast Monitor notify the user that the model training was completed?	@ Georgi Stefanov	Currently the only way to "notify" the user is by displaying how old the active model is.

8. User Interface Design

Pages / ... / Completed 3 Jira links



9. Technical Design

The back-end should send a GET request to

`"/model_training/train?client_key=<string>&unit_key=<string>"`

in order to start model retrain.

10. Deployment and operations

11. Test Approach

11.1. Test Design

No.	Test condition	Testtype (UI, service, integration, manual)	Test conducts with Story no. / Feature	Comments	Result
OS7	The system should be able to provide a way (possibly via button click) to	Manual	1 and 2	TSP	✓

Pages / ... / Completed 3 Jira links

	for one specific pipeline				

12. Stories and Other Activities

12.1. Story 1: Create a new endpoint for triggering model retrain on FS

~~CSL-12833~~ - Story 1: Trigger model retrain for a certain unit CLOSED

12.1.1. Completion criteria

12.1.2. Test approach

12.1.3. Estimate

12.2. Story 2: Create the retrain button and function

~~CSL-12834~~ - Story 2: Create the retrain button and function CLOSED

Create the retrain button in the model performance page and add a function to the data service that send request to the back-end;

The service function should look like so: retrainModel(client_id: number, unit_id: number) and the button should follow the UI draft.

12.2.1. Completion criteria

As a user I can select a unit and then trigger a retraining by clicking on the button in the lower right corner

12.2.2. Test approach

Unit test. The button should trigger a request to the correct endpoint on the back-end

12.2.3. Estimate

12.3. Other activity 1: <Name>

12.3.1. Completion criteria

12.3.2. Estimate

13. Background Information

Reference (e.g. DTS, link,...)	Alias	Title

Pages / ... / Completed 3 Jira links

Reference (e.g. DTS, link,...)	Alias	Title