__HANDOVER__

Foreword
There are 3 models I researched on within the last 6 months of my work at GM&T.

All 3 models are contained in the library GMT-FOM-GAS-RESEARCH which can be found at GM&T tfs repository https://tfs/tfs/Shared/FOM/_git/gmt-fom-gas-research.

Please see this page: GMT-FOM-GAS-RESEARCH for more information about the software you need to have installed on your PC in order to use this library.

The library employs a Python-based task-scheduler which runs the below mentioned models on a schedule.

An example of how to run the scheduler can look like this:

C:
set PATH=%PATH%;C:\Users\ashubert\Anaconda3\condabin
call conda activate gas_research
set PYTHONPATH=C:\git\gmt-fom-gas-research
cd C:\git\gmt-fom-gas-research
python -m main
pause

All three models have corresponding folders at \\trading1\Common\gasmodels

Some of them also have the corresponding tables at GasMongo Mongo Db, pls. see further: GasMongo Database collections | GasModels Mongodb Connection Details

Notes on the library code structure:

- gmt-fom-gas-research\jupyter -this folder contains exemplar Jupyter Notebooks which demonstrate the way the models can be called
- gmt-fom-gas-research\logs - this folder contains txt-based logs
- gmt-fom-gas-reseach\sandbox - this folder contains a number of subfolders, the idea here was for every analyst to have a dedicated subfolder where he/she can keep all the research-based notebooks/scripts

Git Storage Model
This model is concerned with the optimal amounts of storage injections/withdrawals and was borrowed from https://github.com/cmdty/storage and https://pypi.org/project/cmdty-storage/0.1.0/

The two Python packages, that need to be installed are cmdty_storage and curves (using pip).

All numerical computations are done in C# with Python code wrapped around it.

The documentation/papers describing the model are stored at: \\trading1\Common\gasmodels\git_storage\papers

The model has not yet been put into production, rather it's in a pre-production (job-based) mode. As such we've organized to put all the files needed at \\trading1\Common\gasmodels\git_storage\production.

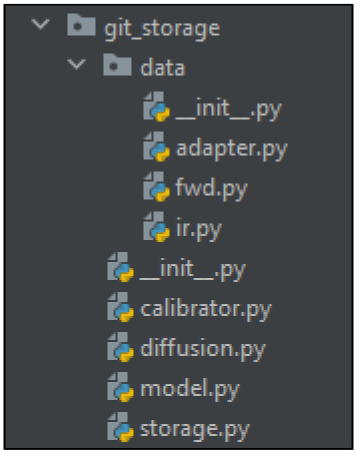
In there

- calibrator.json - contains all the necessary parameters for doing the LSMC regressions (most likely it won't require any changes)
- diffusion.json - contains the values for mean reversion, and 3 volatilities, which defined the SDE of the forward diffusion (this needs to be either calibrated to Big Bird model or being expressed as traders' view)
- storages - this folder contain all the definition of the storages. Each storage needs to have a separate file and contain exactly the same columns as the other files. The storages can be of simple or ratchet type

The Python code in the GMT-FOM-GAS-RESEARCH (see above) library is organized as follows:

- gmt-fom-gas-research\gmt\fom\gas\shared\models\git_storage - contain all the model related scripts
- gmt-fom-gas-research\gmt\fom\gas\shared\jobs\git_storage.py - contains the job (an example of how the model can be called), which is subsequently run in main.py on a schedule.

The code structure is as follows:



Please also have a look at the gmt-fom-gas-research\jupyter\Git Storage Model.ipynb Jupyter Notebook for an example of how the git-storage model can be used in the notebook, you can also see there a method for how the results can be retrieved from MongoDB.
Short Term Power Burn Model
The Short Term Power Burn Model is an attempt to forecast the amount of gas is used to produce electricity (in the UK). Please have a look at this page Short term power burn forecasting for more details about the project.

As an outcome the research phase we've decided to use SARIMAX (1,1,1) x (1,0,1,7) model with the external covariates to be weather (temperature, precipitation, wind) and dark spreads.

All the research-related code can be found in the GMT-FOM-GAS-RESEARCH library (see the foreword above) in the folder sandbox/alexs /short_term_power_burn. In there you can also find various candidate models that we've tried.

The model has a dedicated server folder: \\trading1\Common\gasmodels\short_term_power_burn

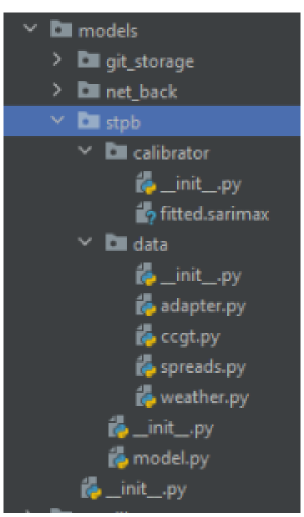
Data:

There are two regimes you can run the model: calibration or forecasting.

In the calibration regime you need to provide *actual (historical)* values for all the covariates and (!) dependent variable. In the forecasting regime you need to provide forecasted values for all the covariates.

- weather:
  - historical values were taken from an external api and saved into the csv files. These files contain temperature, precipitation, wind values for a number of the cities in the UK
  - forecasted values were implemented in ANGARA and need to be retrieved from there (this hasn't been yet implemented)
- spreads
  - historical values are supposed to come from ARC, however we didn't manage to reconcile the output of the the underlying SQL query and the results retrieved using ARC Python api, thus saved the SQL query results into the csv file
  - forecasted values are also supposed to come from ARC (this hasn't been yet implemented)
- ccgt (dependent variable)
  - historical values were provided by Chris Arnold. This time series is only needed for the calibration regime

Code structure:



The calibrator folder contains fitted.sarimax file which is the file produced by the `statsmodels` Python package. The idea is in case the user doesn't need re-calibrate the model, she can re-instantiate the model from this file and produce forecasts. Please bear in mind however, that that `statsmodels` package is using `pikle` Python package underneath to create such a file, and it might not work correctly once run on a new PC. Therefore you might need to re-calibrate the model first prior to running forecasts.

The model is linked to a Python-based scheduler. The job, written in the gmt-fom-gas-research\gmt\fom\gas\shared\jobs\stpb.py file provides a clear an concise demonstration of how to run the model (both in the calibration and forecasting regimes).

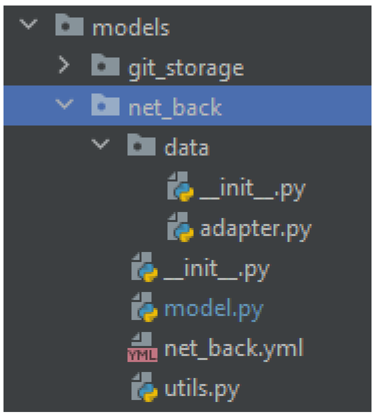
NetBack script

The netback script is not a statistical model per-se, rather an attempt to productionalize the Python script developed by Chris Arnold. Please have a look on Netback model (LNG) for a bit more info around the project.

The script takes two sets of data: *costs* and *variables.* They come in a csv file and were provided by Chris Arnold. Subsequently they were transferred into MongoDB collections.

The code body of the model is the set of transformations of these two data-sets(costs and variables) enriched by the data retrieved from ARC (a number of curves).

Code structure:



The adapter.py file contains DataAdapter class which handles all operations with in and out MongoDB.

The model.py file contains main NetBackModel class as well as NetBackSymbols class.

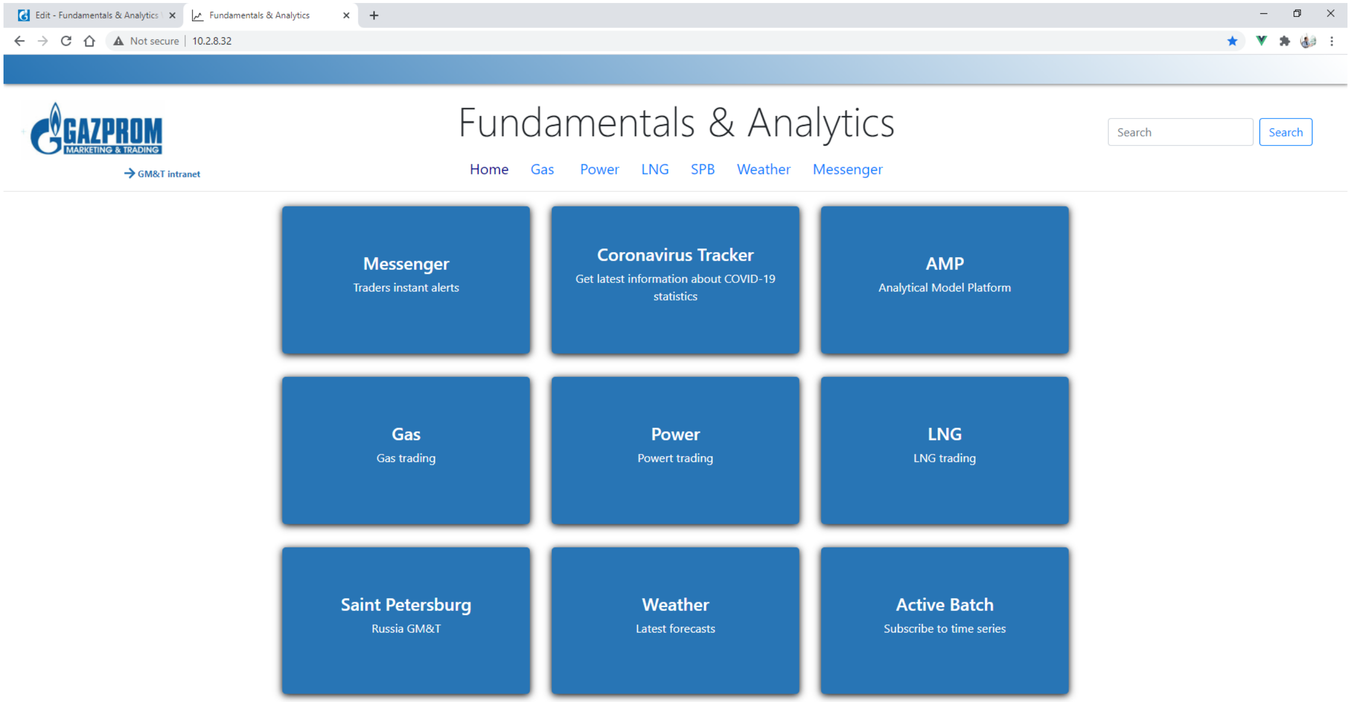
Please note the Mapping class, which is embedded into NetBackModel class. Given a vast amount of constants used in the script, we've decided to collect them all in this Mapping class.

The script is also linked to a Python-based scheduler. The job, written in the gmt-fom-gas-research\gmt\fom\gas\shared\jobs\net_back.py file provides a clear an concise demonstration of how to run the model. The results are stored in the MongoDB (GasMongo) and can be easily retrieved by using this Jupyter Notebook: gmt-fom-gas-research\jupyter\NetBackModel.ipynb.

FOM Web-portal And Trading Alert Messenger

DECOMISSIONED: FOR THE CENTRAL IT TO DEVELOP

Interface



Structure
The portal is powered by the Python (Fast API) backend with the UI written in Vue.js (2x).

Repository: https://tfs/tfs/Shared/FOM/_git/gmt-fom-gas-dashboard

Folder structure:

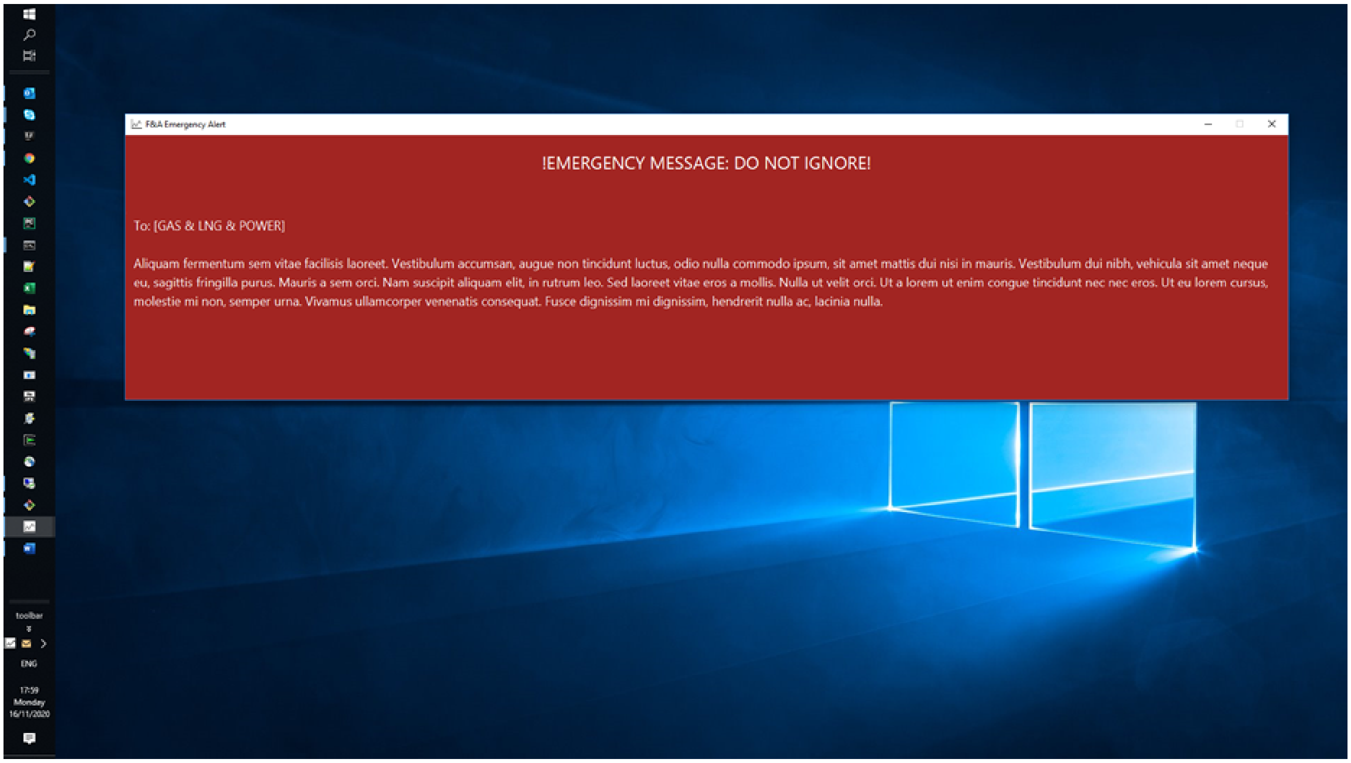
- gmt-fom-gas-dashboard
    - gmt
        - fa
            - lib (for auxiliary python functionality)
            - ui (web-portal)
                - fstapi (web-portal python backend)
                - vue (web-portal Vue frontend)
    - java
        - dashboards
            - messenger (java-based messenger application)
            - Deployment (java classes to be deployed)
            - Install (files to produce setup.exe - 7z self-extracted archive)

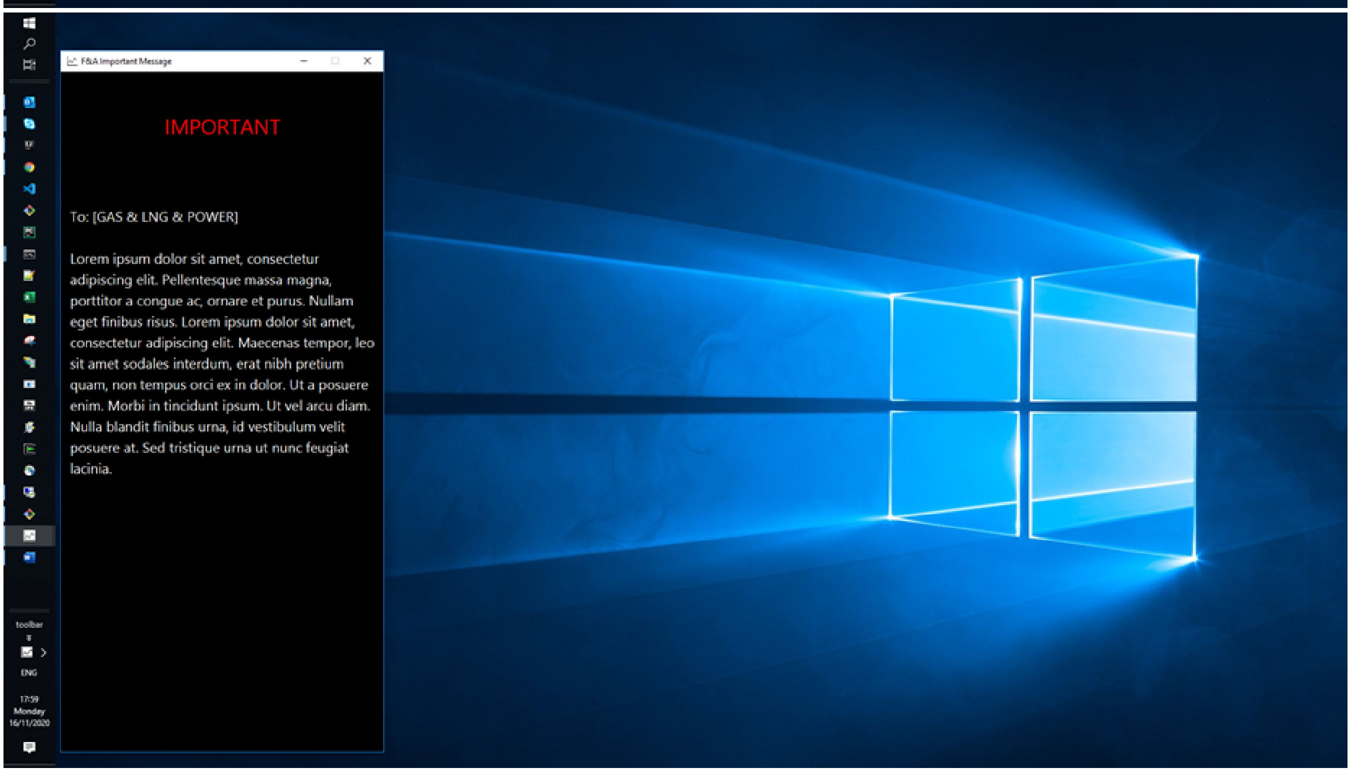
Server structure:

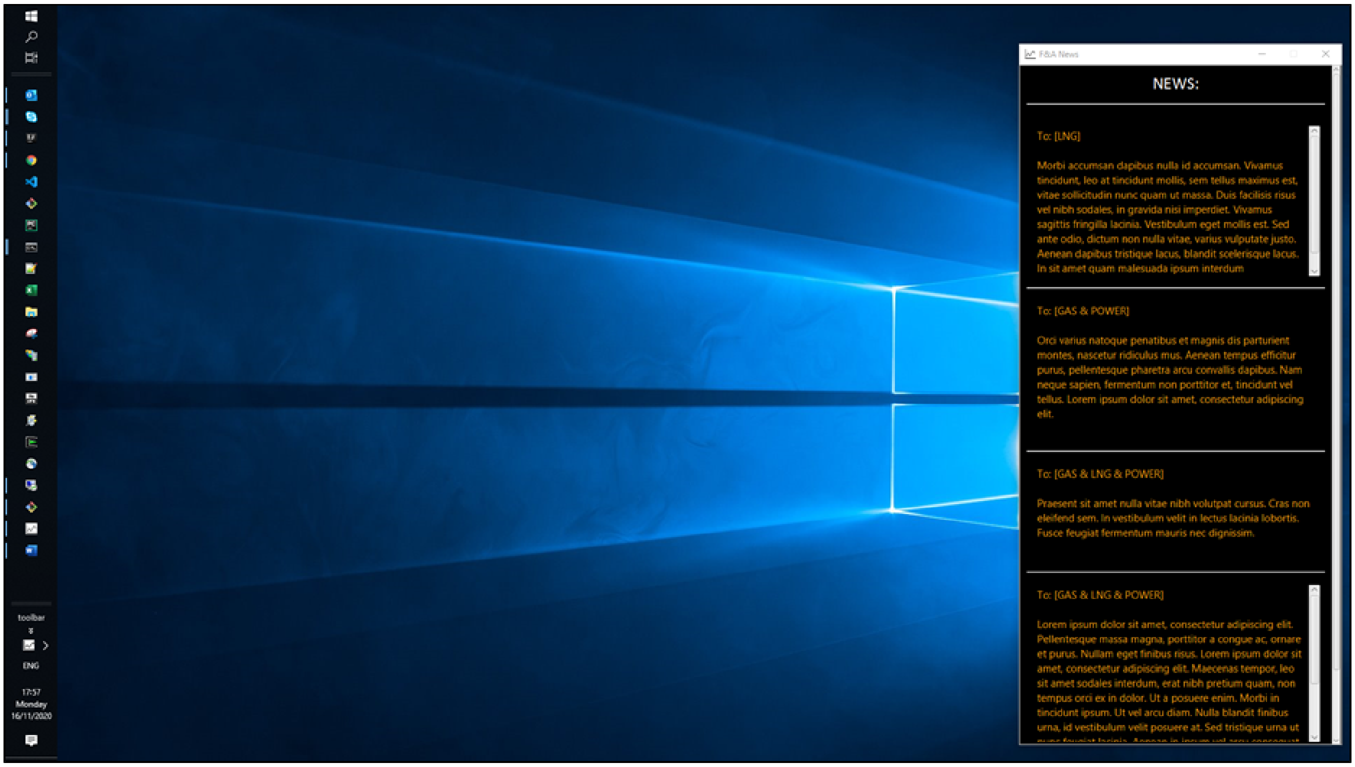
The project is still under development and resides at the gmt00526.gazpromuk.intra

- DEV_PROD: http://10.2.8.32/
- DEV_UAT: http://10.2.8.32:8081/

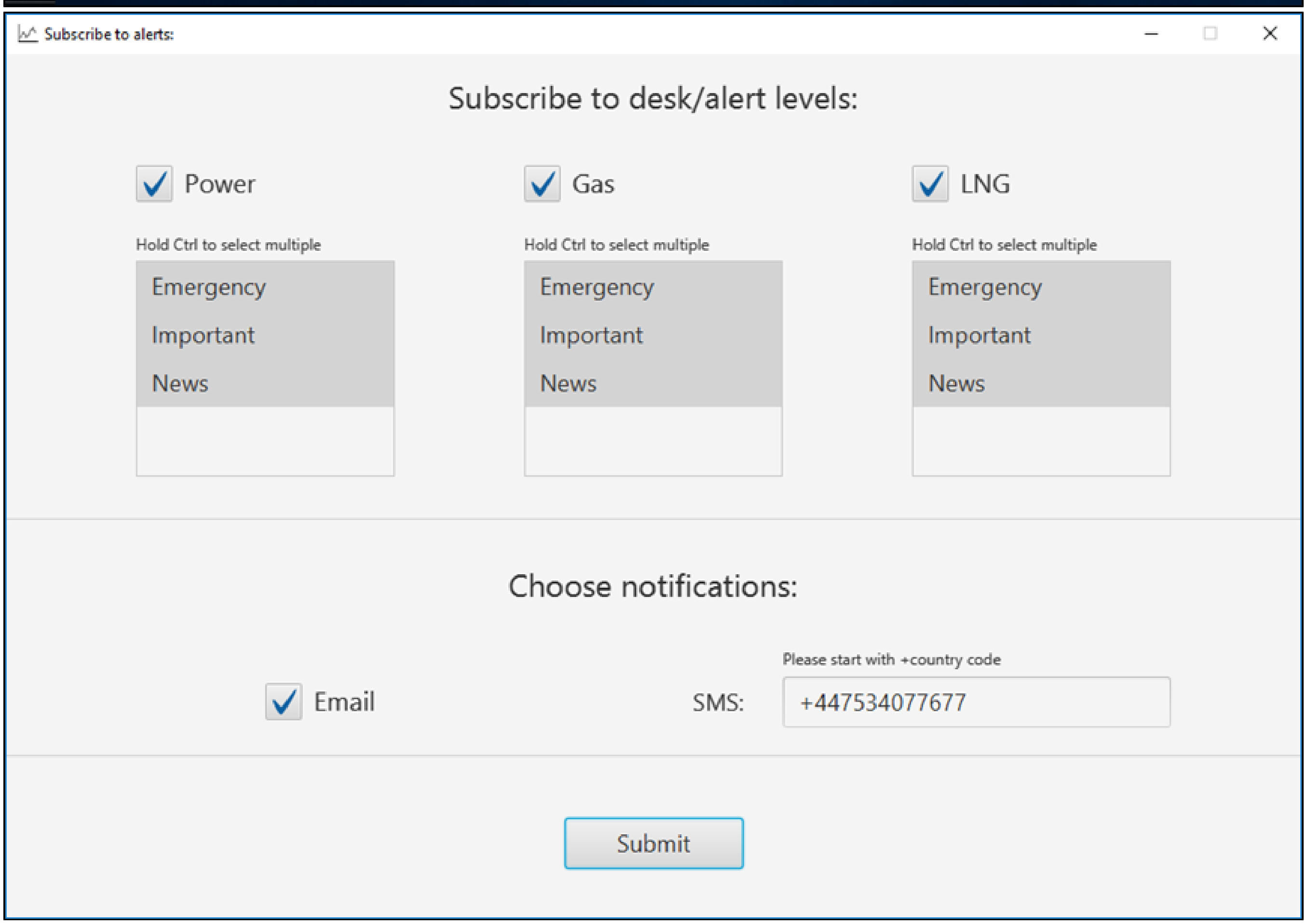
Trader Messenger App
Interface


F&A Emergency Alert
!EMERGENCY MESSAGE: DO NOT IGNORE!
To: [GAS & LNG & POWER]
Aliquam fermentum sem vitae facilisis laoreet. Vestibulum accumsan, augue non tincidunt luctus, odio nulla commodo ipsum, sit amet mattis dui nisi in mauris. Vestibulum dui nibh, vehicula sit amet neque eu, sagittis fringilla purus. Mauris a sem orci. Nam suscipit aliquam elit, in rutrum leo. Sed laoreet vitae eros a mollis. Nulla ut velit orci. Ut a lorem ut enim congue tincidunt nec nec eros. Ut eu lorem cursus, molestie mi non, semper urna. Vivamus ullamcorper venenatis consequat. Fusce dignissim mi dignissim, hendrerit nulla ac, lacinia nulla.
toolbar
ENG
17:59
Monday
16/11/2020

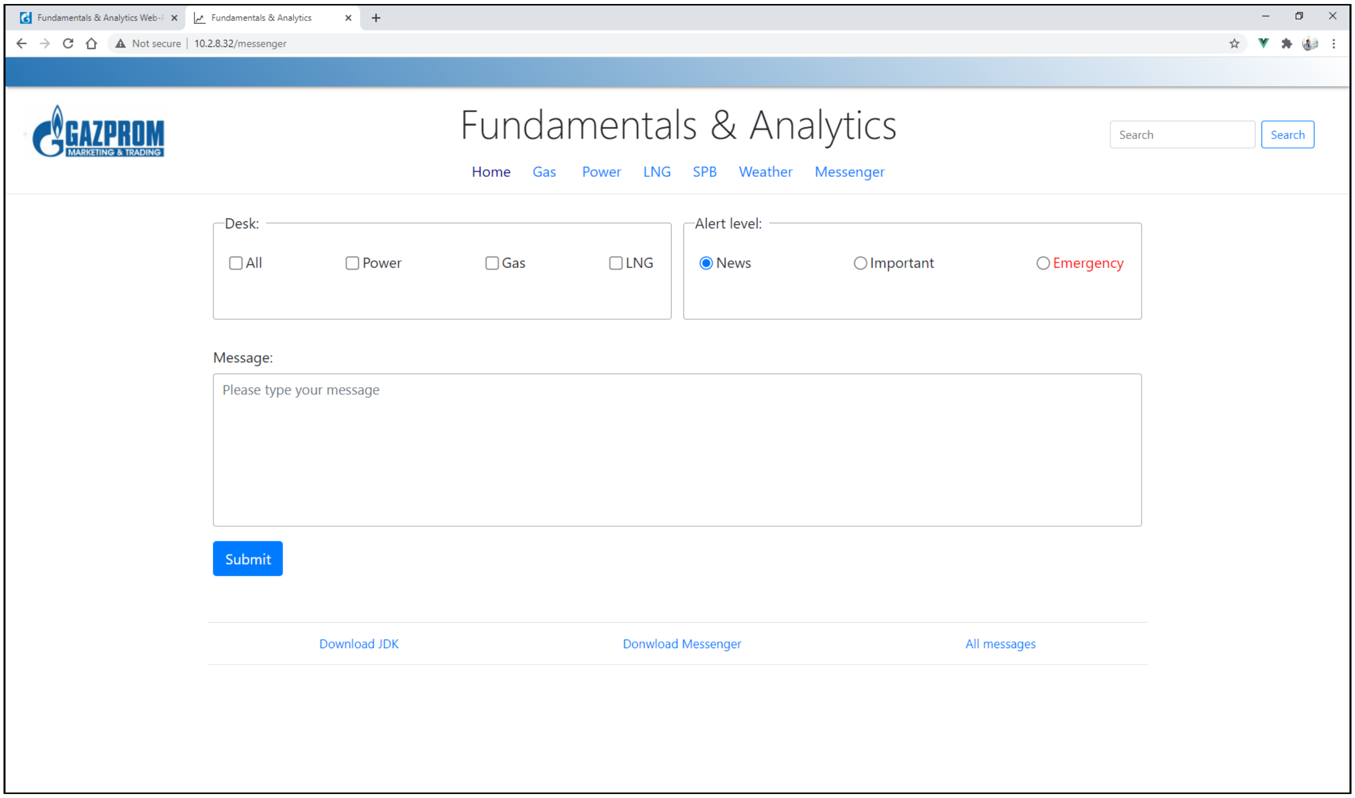

F&A Important Message
IMPORTANT
To: [GAS & LNG & POWER]
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque massa magna, porttitor a congue ac, ornare et purus. Nullam eget finibus risus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas tempor, leo sit amet sodales interdum, erat nibh pretium quam, non tempus orci ex in dolor. Ut a posuere enim. Morbi in tincidunt ipsum. Ut vel arcu diam. Nulla blandit finibus urna, id vestibulum velit posuere at. Sed tristique urna ut nunc feugiat lacinia.
toolbar
ENG
17:59
Monday
16/11/2020

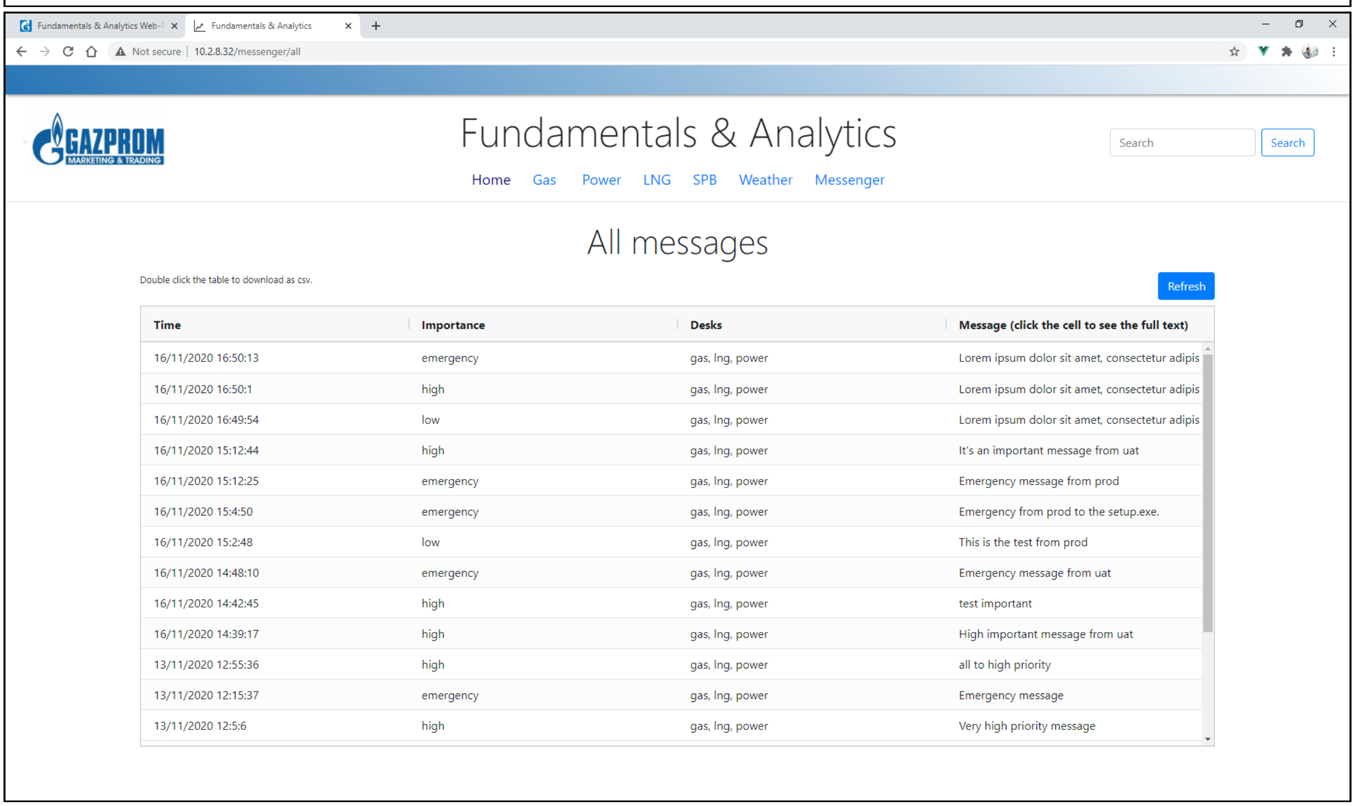


F&A News
NEWS:
To: [LNG]
Morbi accumsan dapibus nulla id accumsan. Vivamus tincidunt, leo at tincidunt mollis, sem tellus maximus est, vitae sollicitudin nunc quam ut massa. Duis facilisis risus vel nibh sodales, in gravida nisi imperdiet. Vivamus sagittis fringilla lacinia. Vestibulum eget mollis est. Sed ante odio, dictum non nulla vitae, varius vulputate justo. Aenean dapibus tristique lacus, blandit scelerisque lacus. In sit amet quam malesuada ipsum interdum
To: [GAS & POWER]
Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aenean tempus efficitur purus, pellentesque pharetra arcu convallis dapibus. Nam neque sapien, fermentum non porttitor et, tincidunt vel tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
To: [GAS & LNG & POWER]
Praesent sit amet nulla vitae nibh volutpat cursus. Cras non eleifend sem. In vestibulum velit in lectus lacinia lobortis. Fusce feugiat fermentum mauris nec dignissim.
To: [GAS & LNG & POWER]
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque massa magna, porttitor a congue ac, ornare et purus. Nullam eget finibus risus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas tempor, leo sit amet sodales interdum, erat nibh pretium quam, non tempus orci ex in dolor. Ut a posuere enim. Morbi in tincidunt ipsum. Ut vel arcu diam. Nulla blandit finibus urna, id vestibulum velit posuere at. Sed tristique urna ut
toolbar
ENG
17:57
Monday
16/11/2020

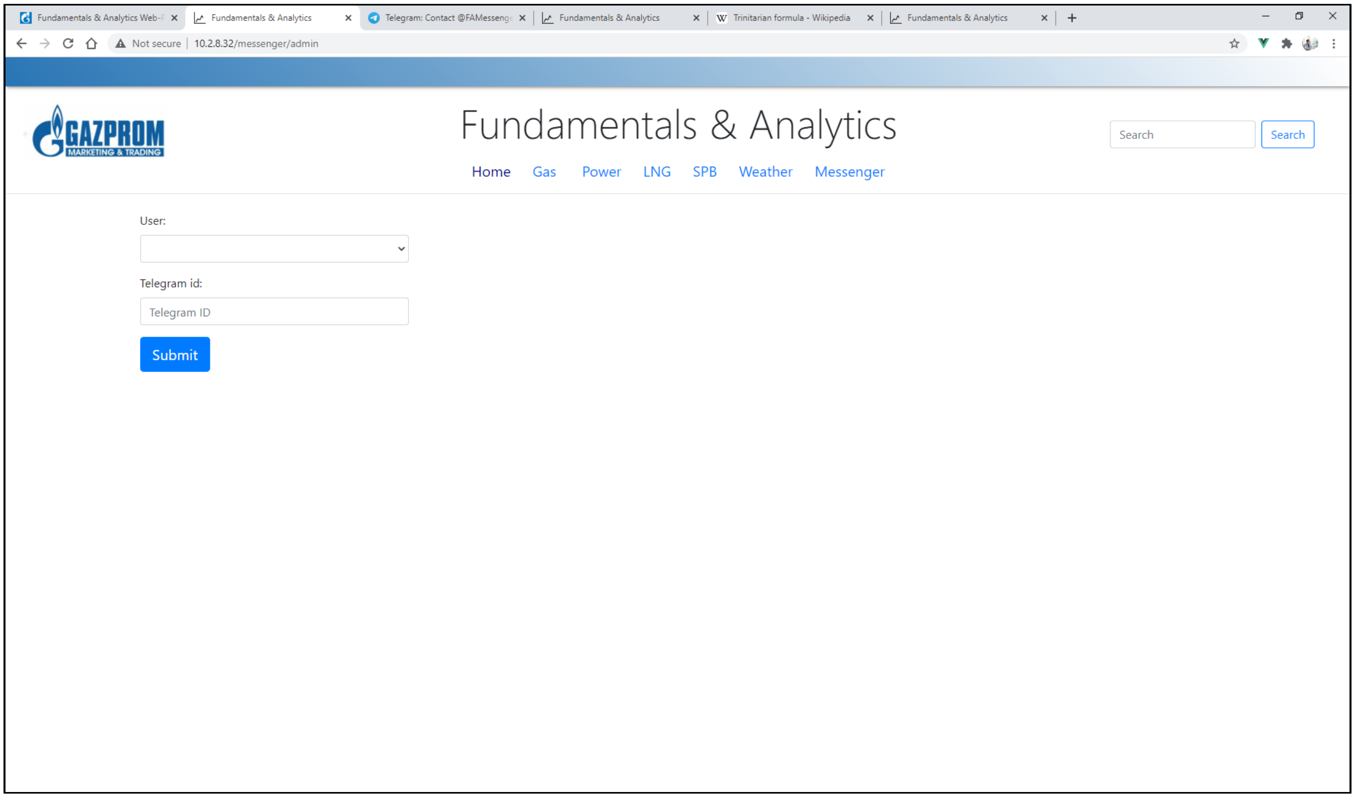


Subscribe to alerts:
Subscribe to desk/alert levels:
Power
Gas
LNG
Hold Ctrl to select multiple
Emergency
Important
News
Hold Ctrl to select multiple
Emergency
Important
News
Hold Ctrl to select multiple
Emergency
Important
News
Choose notifications:
Please start with +country code
Email
SMS:
+447534077677
Submit

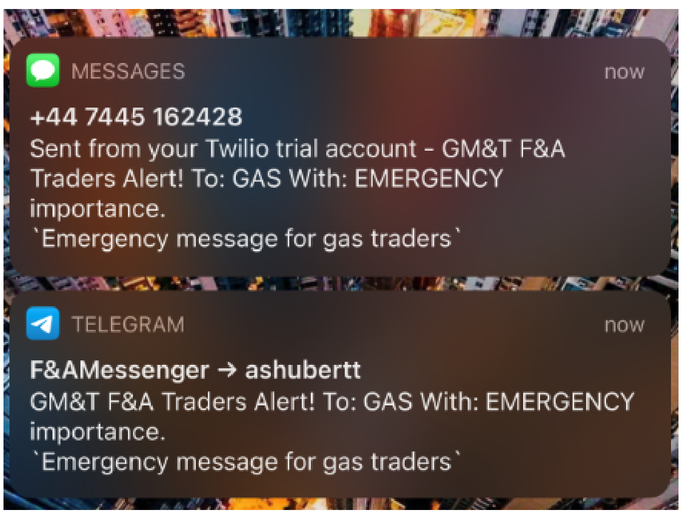




Time	Importance	Desks	Message (click the cell to see the full text)
16/11/2020 16:50:13	emergency	gas, lng, power	Lorem ipsum dolor sit amet, consectetur adipis
16/11/2020 16:50:1	high	gas, lng, power	Lorem ipsum dolor sit amet, consectetur adipis
16/11/2020 16:49:54	low	gas, lng, power	Lorem ipsum dolor sit amet, consectetur adipis
16/11/2020 15:12:44	high	gas, lng, power	It's an important message from uat
16/11/2020 15:12:25	emergency	gas, lng, power	Emergency message from prod
16/11/2020 15:4:50	emergency	gas, lng, power	Emergency from prod to the setup.exe.
16/11/2020 15:2:48	low	gas, lng, power	This is the test from prod
16/11/2020 14:48:10	emergency	gas, lng, power	Emergency message from uat
16/11/2020 14:42:45	high	gas, lng, power	test important
16/11/2020 14:39:17	high	gas, lng, power	High important message from uat
13/11/2020 12:55:36	high	gas, lng, power	all to high priority
13/11/2020 12:15:37	emergency	gas, lng, power	Emergency message
13/11/2020 12:5:6	high	gas, lng, power	Very high priority message

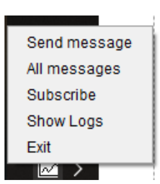


SMS and Telegram Notification:



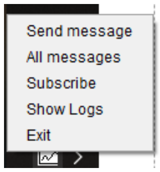
Functionality
When launched for the first time, the user is presented with the subscription window, where she can subscribe to different alert levels for different desks. Subsequently she can subscribe to receive email and/or SMS. In the present configuration, the email and SMS are sent for Important and Emergency alert levels. The user can also opt to receive messages in the Telegram bot.

When the new message arrives it's either shown in Emergency window (which is displayed on top of all windows on *all monitors*), or Important window (which is displayed on top of all windows on the *main* monitor), or pushed as a stack onto the News window (the news window is not going to steal focus, but the standard window alert will be displayed inviting the user to check his news feed). If the user tries to close News window it will be minimized to the taskbar. In order to close News window, the user has to unsubscribe from News alerts for all desks.

From the application tray icon 

context menu the user can open the web-page to send the message or the web-page to see all messages sent so far. She can also see the logs (to send to the support in case the program malfunction) and amend/create her subscription settings.

Python
Python backend: gmt-fom-gas-dashboard\gmt\fa\ui\fstapi\dashboards\messenger
UI (Web)

From within the app context menu 

the user can be forwarded to the following pages on the F&A Web-portal:

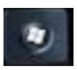
- send message form
- all messages table

There is also a dedicated admin page to update user's telegram ID in case she opts to receive the messages also in the Telegram bot.
UI (JAVA)
The source code consists of an Icon Tray class, 4 JavaFX windows (Emergency, Important, News, Subscribe); Config, Utility, SubscribeStore auxiliary classes and Dispatcher class which governs the flows in the application.

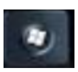
The applications establishes web-socket connection the Python back-end, which in turn broadcasts messages to the users (given their subscription settings) upon the message been sent from the web-page.
UAT Installation
Currently the JAVA 15 runtime is required, which can be downloaded as part of the JDK here.

The messenger setup.exe can be downloaded here.
Python Workshop
Foreword
In what follows I will be writing the sequence of commands using the arrow

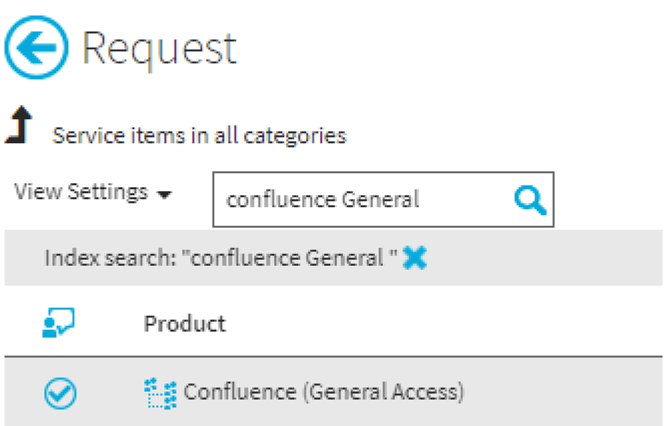
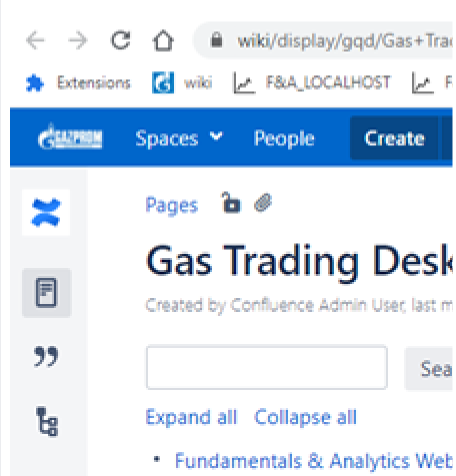
For example:

 + R cmd conda --version

Means:

- Press  button and **R** button (to open Run window)
- Type **cmd** (to open cmd window)
- Type **conda --version**

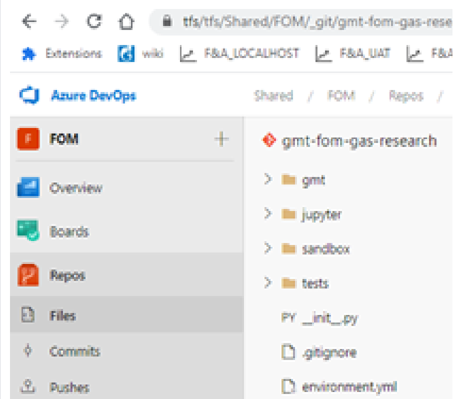
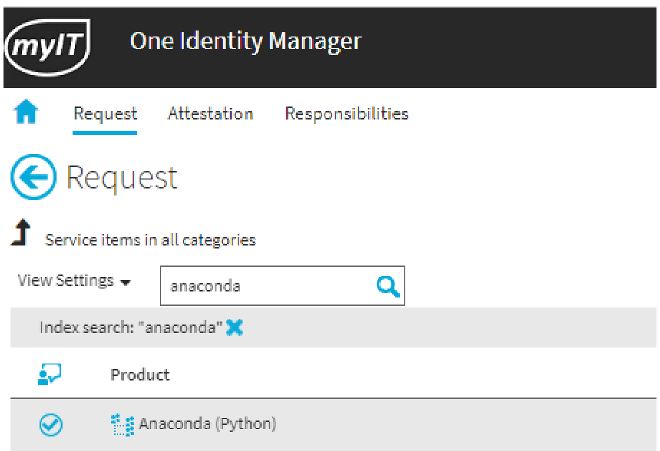
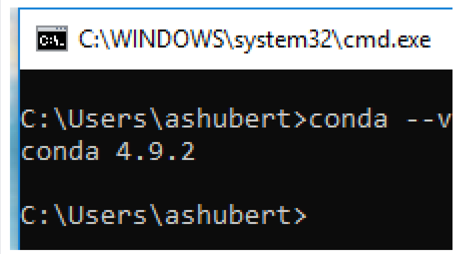
Software To Install

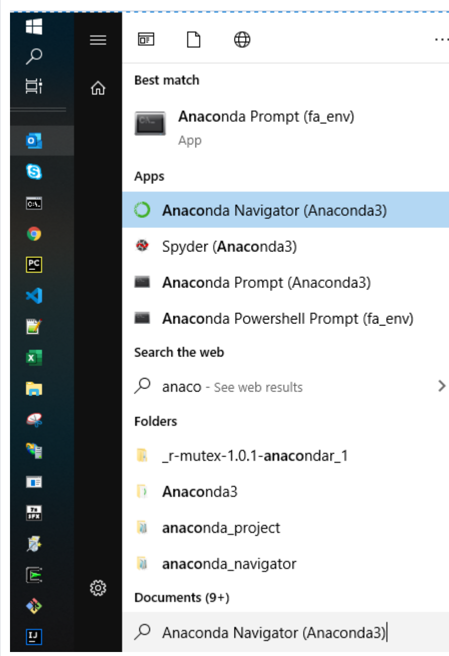
Software	**Request**	**Test**
Access to confluence	 	You can open https://wiki/display/gqd/Gas+Trading+Desk+Quant 

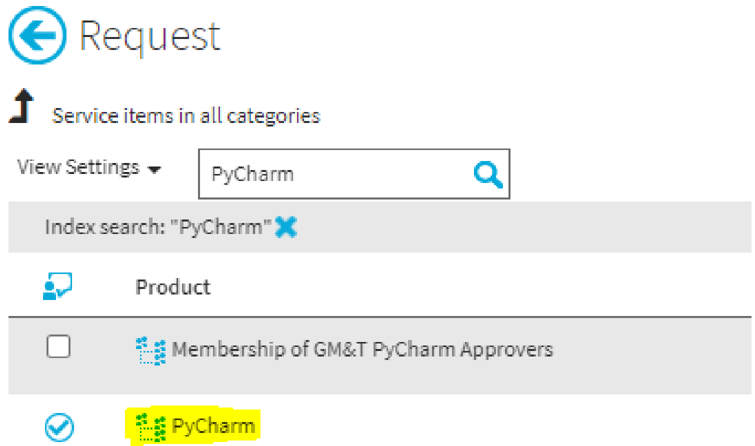
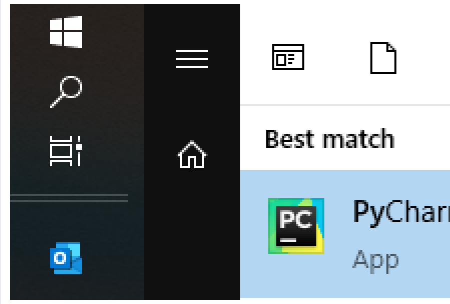
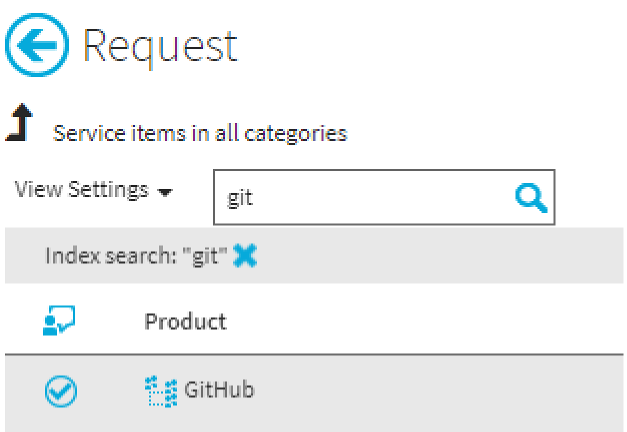
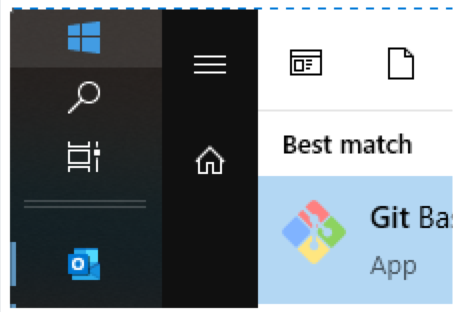
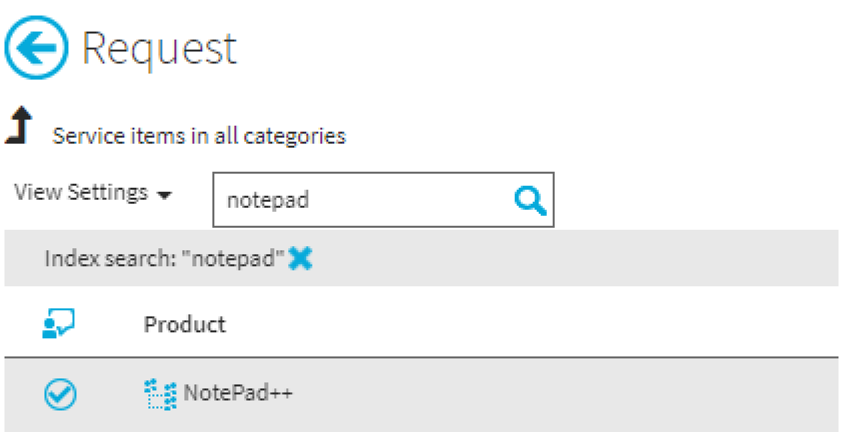
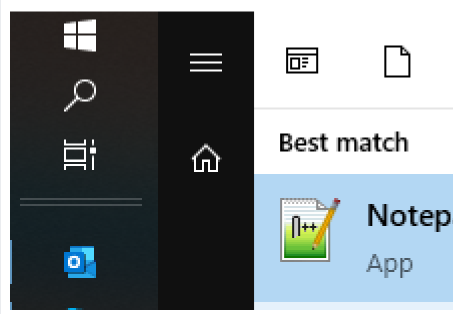
<table>
<tr>
<td>Access to code TFS repository</td>
<td>Email to Ben Hegarty <ben.hegarty@gazprom-mt.com> to give you an access to

FOM/_git/gmt-fom-gas-research repository</td>
<td>You can open:

https://tfs/tfs/Shared/FOM/_git/gmt-fom-gas-resea


</td>
</tr>
<tr>
<td>Anaconda</td>
<td>
</td>
<td>

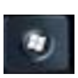
alternatively you have Anaconda Navigator in Star

</td>
</tr>
</table>

PyCharm		Start Menu: 
GitBash		Start Menu: 
Notepad++		Start Menu: 

Running Python

The standard Python distribution consists of an interpreter and a number of packages (standard library). However, for scientific computations the most popular Python distributive is Anaconda.

It might be already the case that you have Python installed at GM&T. To check it:

-  + R cmd   where python

In case it's not installed do the following:

- go to MyIT requests and start a new request
- in the search box type *Anaconda*
- once request is completed, go to Windows Start menu and find Software Center
- in Software Center go to Applications, find Anaconda and proceed with the installation

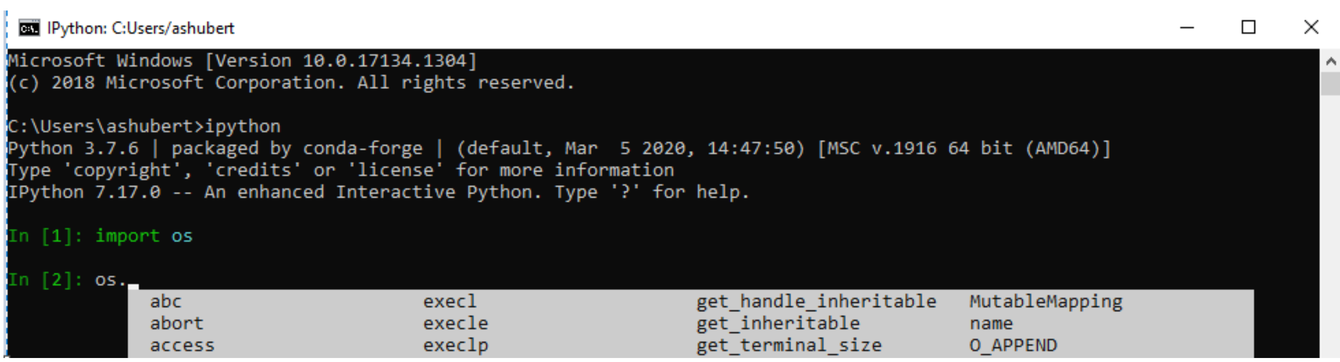
Once Anaconda is installed:

- + R cmd conda --version

Programming IDEs
There are a number of programming IDEs (Integrated Developer Environment) to make programming in Python as comfortable as possible:

CMD:

- you can type your program directly in the **Python interpeter:**
  - + R cmd python print("Hello World")
  - type *exit()* to exit Python
- a bit more user-friendly command line interface is **IPython** (interactive Python):
  - + R cmd ipython
  - now type: *import os* press enter, then type ***os.*** (mind the dot) and press Tab button - you'll see IPython suggests further alternative



Notepad:
You can use any simple text editor (but not MS Word) to type your Python code, including Windows Notepad

- in the folder let's say C:\temp open notepad (+ R notepad), type *print("Hello, world!")* and save as "myscipt.py"
- + R cd C:/temp python myscript.py

PyCharm:

- professional IDE developed by JetBrains. There are two versions: Professional and Community (free).
- to request go to MyIT requests and start a new request
- in the search box type *PyCharm*
- once request is completed, go to Windows Start menu and find Software Center
- in Software Center go to Applications, find Anaconda and proceed with the installation

Jupyter

- this is a web-based IDE which I would highly recommend for learning Python and prototyping
- it is a part of Anaconda Python distribution
- to test that you have jupyter installed run the following:
    - + R cmd   jupyter --version
- to run jupyter notebook:
    - + R cmd   cd <your folder> jupyter notebook

Anaconda
To be able to "preserve" the state of packages you used in your current application you need to create a *virtual environment.* Physically it's just a folder on your PC where all the packages are going to be saved.

Anaconda has it's own package manager `conda` which allows you to easily create a new virtual environment. In order to to do it type the following in the + R cmd:

- *conda create --name <name of the environment> python=3.7*

Once it's done type:

- *conda activate <name of the environment>*

To see all the existing environments on your machine type:

- *conda info --envs*

To deactivate the environment, type:

- *conda deactivate*

To install a new package inside your environment, once the environment is activated, type (for e.g. pandas package):

- *conda install pandas*

To learn mode *conda* commands read Conda Cheat Sheet.
Jupyter Notebook

- Keyboard Shortcuts
- Markdown Cheat Sheet
- Simple LaTex commands
- YouTube shortTutorial | longTutorial

Software Development Life Cycle (SDLC)
GIT
Introduction to git
Azure DevOps Server (TFS)
Microsoft Documenation
AMP
Documentation WikiPage: AMP (Analytics Modelling Platform)
Python Lessons
Internal Workshop:

- Python Workshop**:** jupyter | html
- An example of a modelling class: jupyter | html
- Exercises: html
- Solutions: html

External resources:

- YouTube Python Course
- Simply Python Tutorial

Python package-specific tutorials

- numpy
- scipy
- pandas
- matplotlib

- seaborn
- statsmodels
- scikit-learn
- prophet
- tensorflow
- keras

Statistical Modelling in Python

	Description	Files	Data Set	conda environment	Comments

1	General time-series modelling framework in Python (based on **ARIMAX** model):
	• data retrieval
	• exploratory analysis and visualisations
	• data pre-processing
	  ◦ sample size
	  ◦ imputations
	  ◦ stationarity
	  ◦ transformations
	• design matrix
	• data-set partitions
	  ◦ training
	  ◦ validation
	  ◦ testing
	• model selection
	  ◦ model parameters specification
	    ▪ heuristic
	    ▪ grid methods
	• forecasting
	• model performance metrics
	• cross-validation error
	• final model calibration
	• out-of-time performance				

2	**Short Term Power Burn Model**
	• an example of a newly developed model for Gas Analytics
	
	• data:
	  ◦ dark, clean spark spreads and weather covariates
	  ◦ is coming from ARC DB and external api
	  ◦ pre-processed and copied to Gas MongoDB
	
	• models:
	  ◦ Linear Regression
	  ◦ Support Vector (kernel) Regression
	  ◦ SARIMAX
	  ◦ LSTM RNN
	
	Additional notes:
	• This article clearly defines *causality* | *correlation* | *forecasting,* for it's essential to clearly define the purpose of the research prior to conducting one.
	
	• Trying to formulate the ultimate target of any statistical research we can try (in the first approach) to reduce it to the following two main tasks :
	  1. *Choose the loss function we'd like to minimize*
	    ▪ This gives rise, among others, to least ordinary squares loss function, but also to a variety of penalized (regularized) loss functions such as (pls see this article for explanation)
	      ▪ *ridge*
	      ▪ *lasso*
	      ▪ *elastic net*
	
	  2. Capture the functional relationship between the dependent variable and covariates
	
	    ▪ This gives rise to the following cascade of modelling frameworks:
	      ▪ Simple linear model
	      ▪ General linear model
	      ▪ Generalized linear model
	      ▪ Generalized additive model
	
	    ▪ The following also might be of interest*:*
	      ▪ Local regression (LOESS | LOWESS)
	      ▪ Polynomial regression
	        ▪ Basis functions / splines regression
	      ▪ Support Vector Regression
	      ▪ Long Short Term Memory Recurrent Neural Network
	        ▪ Description
	        ▪ Tensor Flow tutorial
		• Data pre-processing notebooks
		• Candidate Models
			\\trading1\Common\gasmodels\short_term_power_burn\data		

How To

- Get Data from ARC database: jupyter | html