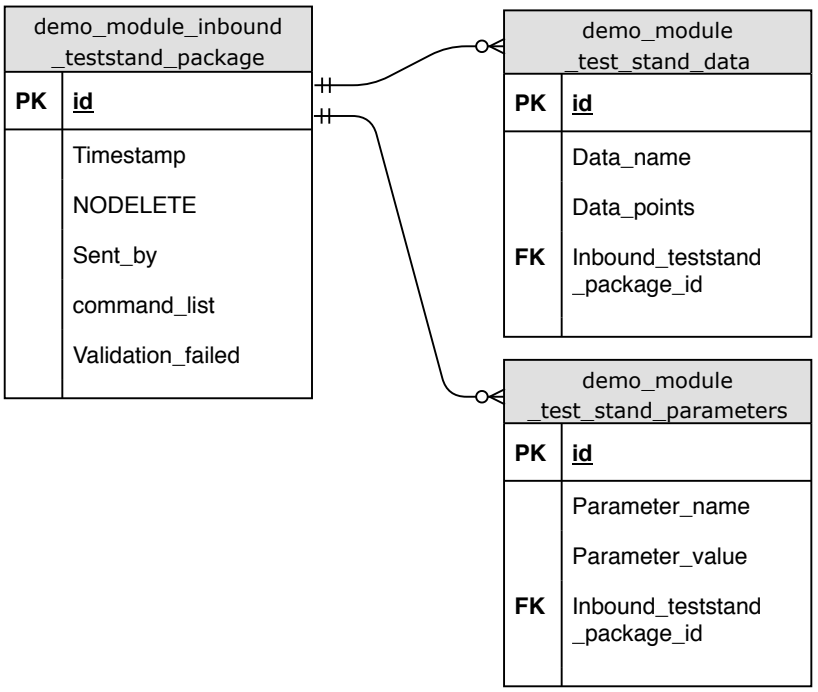
Anvendelse af databasen i Webinterface

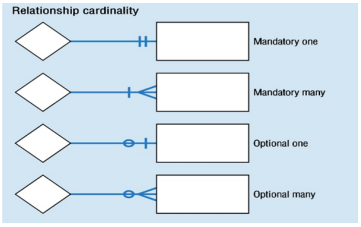
Team 2, april 2020. Ref.: jba.

ERD

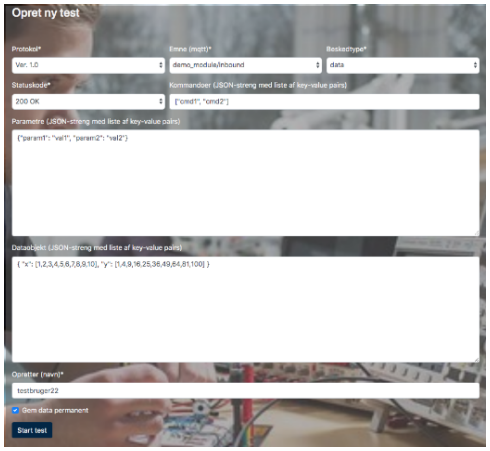
Entity Relationship Diagram (ERD) viser databasens struktur per Timebox 8: d. 5. april 2020.

Der kan stadig komme ændringer til strukturen.





Ét resultat i databasen giver følgende tabeller



	id [PK] integer	Timestamp character varying (200)	NODELETE boolean	Sent_by character varying (200)	command_list character varying[] (20)	Validation_failed boolean
1	1	04/04/20-10:34:56	true	testbruger22	{cmd1,cmd2}	false

	id [PK] integer	Data_name character varying (100)	Data_points jsonb	Inbound_teststand_package_id integer
1	1	x	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	1
2	2	y	[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]	1

	id [PK] integer	Parameter_name character varying (100)	Parameter_value character varying (100)	Inbound_teststand_package_id integer
1	1	param1	val1	1
2	2	param2	val2	1

#	Tid (UTC)	Data		Parametre	
1	Testet af: testbruger22	x	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]	param1	val1
	Den: 04/04/20-10:34:56	y	[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]	param2	val2

Arbejd gerne med databasen i Django via shell først, for at få mere føling med sammenhængene i data og joins...

```
~/projects/e4pro4/server-setup(master) » docker-compose exec webinterface python manage.py shell
Python 3.8.0 (default, Nov 15 2019, 02:22:06)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.10.2 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from demo_module.models import Inbound_teststand_package, Test_stand_data, Test_stand_parameters

In [30]: Test_stand_data.objects.filter(Inbound_teststand_package__Sent_by='testbruger22')
Out[30]: <QuerySet [<Test_stand_data: 04/04/20-10:34:56 - x>, <Test_stand_data: 04/04/20-10:34:56 - y>]>

In [31]: Test_stand_data.objects.filter(Inbound_teststand_package__Sent_by='testbruger22')[0]
Out[31]: <Test_stand_data: 04/04/20-10:34:56 - x>

In [32]: Test_stand_data.objects.filter(Inbound_teststand_package__Sent_by='testbruger22')[1]
Out[32]: <Test_stand_data: 04/04/20-10:34:56 - y>

In [37]: Test_stand_data.objects.filter(Inbound_teststand_package__id=1)[0:2]
Out[37]: <QuerySet [<Test_stand_data: 04/04/20-10:34:56 - x>, <Test_stand_data: 04/04/20-10:34:56 - y>]>

In [42]: Test_stand_data.objects.filter(Inbound_teststand_package__id=1)[0].Data_points
Out[42]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [45]: Test_stand_data.objects.filter(Inbound_teststand_package__Sent_by__contains='bruger')[0]
Out[45]: <Test_stand_data: 04/04/20-10:34:56 - x>

In [67]: Inbound_teststand_package.objects.get(id=1).data.all()
Out[67]: <QuerySet [<Test_stand_data: 04/04/20-10:34:56 - x>, <Test_stand_data: 04/04/20-10:34:56 - y>]>

In [69]: my_dataset = Inbound_teststand_package.objects.get(id=1).data.all()[0]

In [70]: {my_dataset.Data_name : my_dataset.Data_points}
Out[70]: {'x': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]}

In [88]: my_params = Inbound_teststand_package.objects.get(id=1).parameters.all()

In [89]: dict([(p.Parameter_name, p.Parameter_value) for p in my_params])
Out[89]: {'param1': 'val1', 'param2': 'val2'}
```

Modelnavnet (klassen) repræsenterer *hele* tabellen i databasen. En instans af klassen (et objekt) repræsenterer en enkelt række.

- *modelnavn*.objects.all() giver alle rækker i tabellen.
- *modelnavn*.objects.filter(*kriterium*) udvælger rækker, der opfylder kriterium, og giver 0-til-mange resultater.
- *modelnavn*.objects.get(*kriterium*) udvælger én række, der opfylder kriterium, og giver kun 1 resultat eller en fejl.
- Filtre kan sammenkædes: ...objects.filter(*krit1*).filter(*krit2*)...

Et QuerySet er et resultatsæt med 0-til-mange elementer, så man indekserer i det, for at få specifikke resultater ud, fx vælges element [0] i linje 31.

Dobbelte underscores i filter(), get(), m.m. repræsenterer enten *"."* for objekter eller magiske metodekald i Django:

- Se fx linje 32, hvor vi vælger fra *Test_stand_data*-tabellen ud fra at testen skal være sendt af 'testbruger22'.
- Denne metode repræsenterer også et join, fordi *Sent_by*-feltet er i en anden tabel.
- Se linje 45, hvor vi tjekker at testen er sendt af nogen, hvor navnet indeholder 'bruger'
- Data/værdier findes frem ved at tilgå de specifikke feltnavne (se linje 42).

Man kan også genskabe oprindelige relationer vha. Django's automatiske joins (se linje 69). Man kan lave større nye objekter ved at iterere over querysets med fx Python's List Comprehensions (se linje 89).

Se mere her: https://docs.djangoproject.com/en/2.2/topics/db/queries/#lookups-that-span-relationships