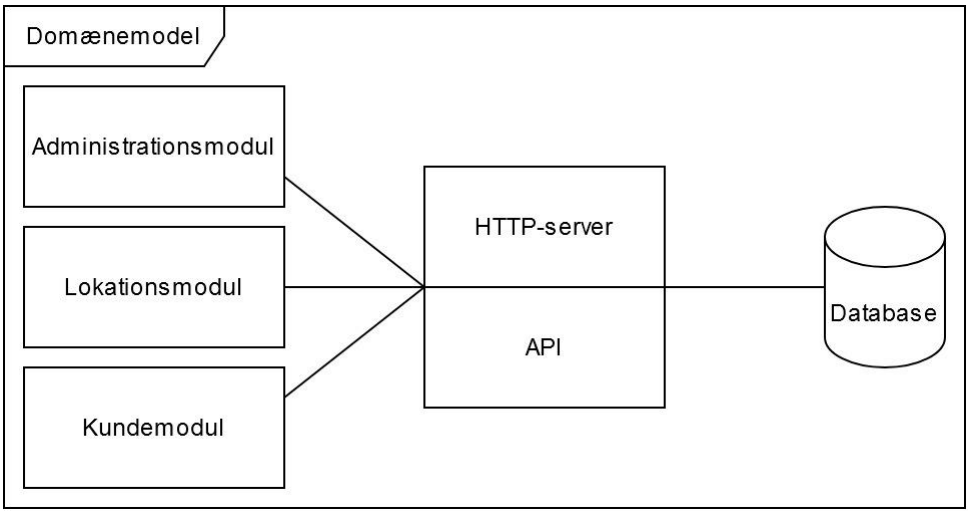
# Internt system til Klik-og-Hent

## Krav for systemet (kopieret fra opgavebeskrivelse)

- Systemet skal bygges i Python, men du vælger selv et eventuelt framework
- Systemet skal både have HTTP-tilgængelige views samt kunne udstille mindst én type API
- Systemet indeholder selv sin egen data, så det skal indeholde datamodeller
- Der skal være et administrations-modul, til f.eks. brugerstyring
- Der skal være et lokations-modul, hvor de ansatte kan modtage og udlevere ordrer
- Der skal være et kunde-modul, hvor kunder kan se status på deres ordrer
- Systemet skal som minimum kunne sende mails til kunderne, f.eks. når deres ordre er klar til afhentning

## Domænebeskrivelse




*Figur 1 domænemodel for klik-og-hent system*

De nuværende krav ligger op til, at der kun er 3 moduler, men der kan godt komme flere i fremtiden. Systemet sættes op som på figur 1, med 2 views i form af de 2 moduler fra kravene, samt en backend, som serverer views, og gør et API tilngængeligt. Backenden har så også en kontaktflade til en database hvor backenden gør brug af- og fremstiller data til views.

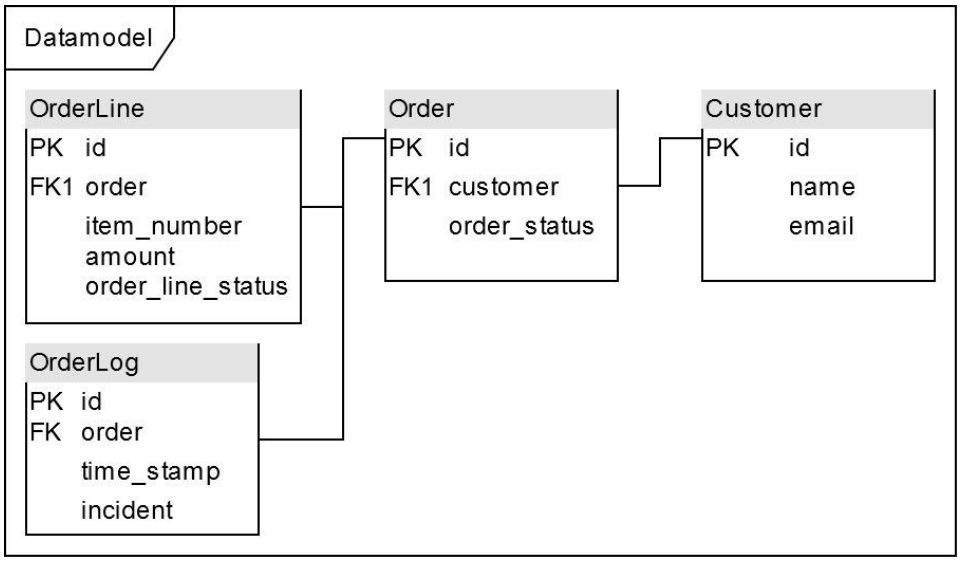
## Filstruktur

Der tages udgangspunkt i brugen af Python, og for at sætte de forskellige webapplikationsfiler op, gøres der derfor også brug af forskellige frameworks. Da frameworks ikke var defineret i kravene, vælges der her Django. Fil strukturen er udarbejdet baseret på autogenereret boilerplate og standard for mappe- og filstruktur

```
klik-og-hent/
├── api/
│   ├── __init__.py
│   ├── serializers.py
│   ├── urls.py
│   ├── views.py
├── base/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── test.py
│   ├── views.py
├── ClickAndCollect/
│   ├── __init__.py
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
├── AdminModule/
│   ├── migrations/
│   ├── templates/
│   │   ├── AdminModule/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   ├── views.py
```

```
├── LocationModule/
│   ├── migrations/
│   ├── templates/
│   │   ├── LocationModule/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   ├── views.py
├── CustomerModule/
│   ├── migrations/
│   ├── templates/
│   │   ├── CustomerModule/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── urls.py
│   ├── views.py
├── db.sqlite3
└── manage.py
```

# Datamodel




*Figur 2 datamodel for systemet*

Den angivne datamodel er baseret på kravene som er beskrevet tidligere under Krav for systemet.

## OrderLine består af

- **id**: et autogenereret id.
- **order**: en foreign key til et Order objekt, så der dannes en N:1 forbindelse til et Order objekt.
- **Item_number**: et varenummer
- **amount**: mængden
- **order_line_status**: statussen for orderline

## Order består af

- **id**: et autogenereret id.
- **customer**: en foreign key til et Customer objekt, så der dannes en N:1 forbindelse til et Customer objekt.
- **orderStatus**: Hvilken status ordrerne er i, om den er behandlet, klar til at blive hentet, hentet, osv. Dette kunne også håndteres med en ENUM.

## Customer består af

- **id**: et autogenereret id.
- **name**: kundens navn
- **email**: kundens e-mailadresse

## OrderLog består af

- **id:** et autogeneret id.
- **order:** en foreign key til et Order objekt, så der kan dannes en N:1 forbindelse ti let Order objekt
- **time_stamp:** tidspunkten hvor loggen laves
- **incident:** hvilken hændelse der er sket

## Fravalgt men muligt brugbare attributter:

- **orderLocation**: Som skal matche lokationen som håndterer ordrerne.

- **orderType**: Hvilken type ordrer det er, i dette tilfælde bliver det nok en Klik og Hent, men da der nok kan komme flere forskellige typer i fremtiden, er dette gjort åbent. En måde at håndtere det flotte kunne være ved brug af et ENUM.
- **orderCreated**: Hvornår ordren er lavet
- **orderFinished**: Hvornår ordren er afsluttet

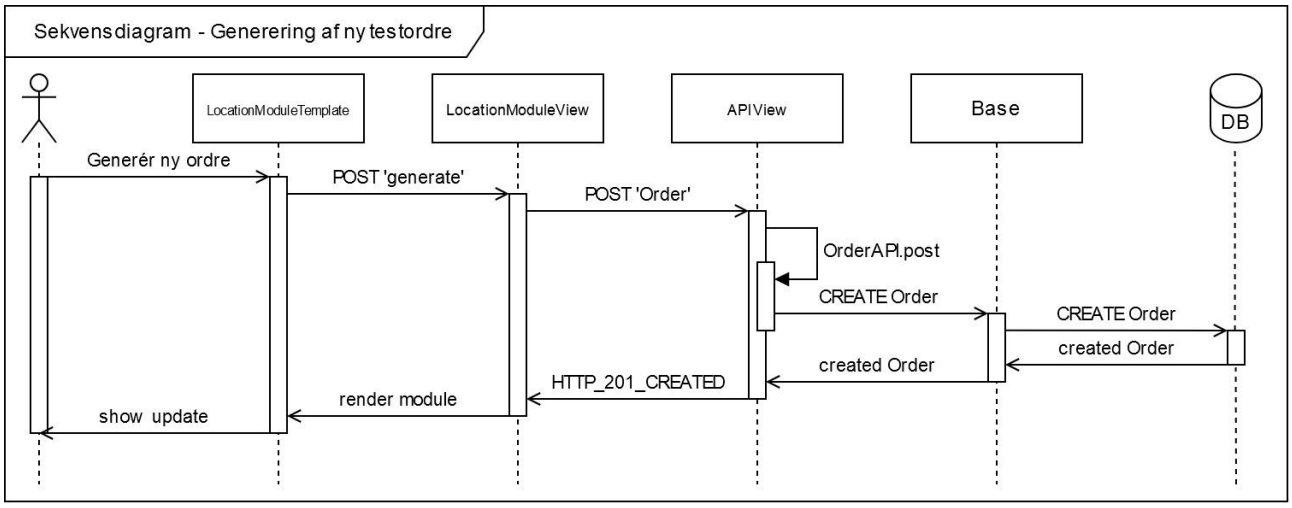
## Minimal viable product – udviklingskrav

- Systemet skal udstille en webapp.
- Der skal være en knap der genererer en test-ordre, med mindst én ordrelinje (som indeholder varenummer og antal).
- Man skal kunne se en liste af alle aktive test-ordrer.
- Man skal kunne åbne en ordre, og se detaljer.
- Man skal kunne "modtage" én ordrelinje af gangen - altså bekendtgøre at denne vare nu er på lager, og er klar til kunden.
- Når alle ordrelinjer er modtaget, skal der sendes en mail til kunden at ordren er klar til afhentning.
- Der skal være en "Udlever" knap på en ordre, som markerer ordren som udleveret.
- Alle aktive handlinger skal logges og kunne ses på en ordre.

### Ikke-krav

- Der skal ikke være nogen brugerstyring eller autentifikation. Vi antager at man er logget ind som en ansat.
- Der behøver ikke være noget grafisk design.
- Data behøver ikke være persistent - det må godt bare ligge i memory.
- Data behøver ikke være gemt i en database.
- Der behøver ikke rent faktisk at blive sendt en mail, så længe koden lader som om.

## Sekvensdiagram




*Figur 3 sekvensdiagram for generering af en ny ordre*

En simpel overfladisk beskrivelse af, hvordan et flow ville fungere til håndteringen af at generere en ny testordre. Det giver et nogenlunde overblik for hvilke moduler som gør sig gældende i et lignende flow.