Opgave: Rejser med PlusBus

# Introduktion

### Formål med opgaven

* Du skal opnå mere rutine med at benytte en GUI og en SQL-database i en Python-program.
* At løse denne opgave forbereder dig godt på eksamen, fordi din eksamensopgave kommer til at ligne PlusBus- og DanskCargo-opgaven.

### Fremgangsmåde

* Læs opgavestillingen nøjagtigt!
* Vi løser stort set de samme problemer som i DanskCargo-opgaven men nu får du ingen detaljerede anvisninger og heller ingen eksempel-kode. Find detaljerne hvordan man gør i DanskCargo-opgaven.
* Kopier skamløst kode fra din DanskCargo-løsning eller fra lærerens sidste DanskCargo-milepæl.
* Kommenter din kode rigeligt så du forstå den stadig når du kommer til at genbruge den. Dine kommentarer vil også være meget nyttige, når du skal forklare din kode til eksamen.
* Push din løsnings-repository mindst to gange om dagen til GitHub: Før frokostpausen og før du går hjem.

### Hvis du sidder fast med noget:

* Læs altid hele opgaven før du begynder at løse den.
* Research selv (google, w3schools.com, stackoverflow.com, …).
* Spørg de andre elever.
* Spørg læreren. Senest når du har siddet fast i en opgave i en halv time, spørg læreren eller send læreren en Teams Chat besked som ”*jeg sidder fast i en opgave*”.

# Problemformulering

PlusBus er et busselskab for busrejser indenfor Danmark. Du skal udvikle et program for PlusBus.

Den skal have en brugergrænseflade til håndtering af oprettelse og vedligeholdelse af kundekartoteket og af de rejsearrangementer kunderne kan booke. Derudover kan man i programmet gemme, redigere og slette bookinger.

# Kravspecifikation

De følgende oplysninger skal gemmes i en SQL-database og håndteres af vores program:

Om rejsebureauets **kunder**:

* Efternavn
* Kontakt (Telefonnummer eller E-mail)

Om de **rejser** som kunderne kan booke:

* Rute (i hvilken by begynder rejsen og i hvilken by ender den, sammenfattet i én kolonne)
* Dato
* PladsKapacitet (for så mange personer er der plads ved denne rejse)

Om **bookinger**:

* KundeId
* RejseId
* Pladser (hvor mange pladser har kunden booket på denne rejse)

Alle nævnte oplysninger kan man oprette, redigere og slette i GUIen. (Sletning kunne være fuldstændigt eller logisk. Du beslutter hvad giver mere mening.)

Før man kan gemme en ny booking skal det tjekkes om rejsen har nok frie pladser tilbage.

# Eksempel for en Løsningsstruktur

Denne løsningsstruktur er ikke fuldstændigt men den giver en god overblik over delopgaverne og deres rækkefølge. Kig i DanskCargo-opgaven for detaljerne.

## Opret filer

Opret en ny mappe kaldet plusbus i pycharm i dit løsningsprojekt.

Opret nu en indledningsvis tom python-fil i denne mappe for hver lag:

* Datastruktur-lag
* Funktionelt lag
* SQL-Database-lag
* GUI-lag

Det er en god idé at også åbne din DanskCargo-solution eller lærerens sidste DanskCargo-milepæl parallelt så du kan sammenligne og genbruge koden.

## Planlæg datastrukturen og databasen

Besvar de følgende spørgsmål. Helst på et stykke papir først.

* Hvilke database-tabeller har du brug for? Hvad skal de hedde?
* Hvilke kolonner har du brug for i tabellerne? Hvad skal de hedde?
* Hvilke (sqlalchemy-)datatyper har kolonnerne?
* Hvad er dine primærnøgle?
* Hvilke forbindelser er der mellem tabellerne (via fremmednøgler)?

Snak med læreren om din udkast før du opretter databasen.

## Skriv og test datastrukturen og databasen

* Definer databasens datastruktur med hjælp af biblioteket sqlalchemy.
* Tilføj metoder til dine data-klasser, som:
  + udskriver klasseobjektet i en læsbar form
  + konverterer klasseobjektet til en tupel
  + finder ud af, om et klasseobjekt indeholder gyldige, ikke soft deleted data
  + konverterer tupler til klasseobjekter
* Definer hvilken slags database du benytter og hvor den ligger.
* Opret testdata, så du kan teste databasen, før du er færdig med at skrive GUI'en.
* Skriv en funktion, som returnerer en liste med alle poster fra en bestemt tabel i SQL-databasen.
* Skriv en funktion, som returnerer kun én post med en bestemt ID fra en bestemt tabel i SQL-databasen.
* Test din database ved at skrive testdata i den og læse og udprinte data fra databasen.

## Planlæg GUI’en

* Tegn en udkast.
* Hvilke widget typer benytter du?
* Hvad er widget-hierarkiet? Dvs., definer for hvert widget hvilken overordnede widget den er tilknyttet.
* Hvilke regler følger dine widget-navne? (Gør copy&paste nem!)

Snak med læreren om din udkast før du opretter GUI‘en.

## Skriv og test GUI’en

* I første omgang skriv bare GUI'en til at administrere kunderne.
* Definer widgets.
* Definer funktioner, der læser, tømmer og fylder dine indtastningswidgets med tekst.
* Forbind dine widgets og funktioner.
* Skriv en funktion, som læser alle poster fra database-tabel og skriver de gyldige (ikke slettede) poster ind i treeviewen.
* Skriv en funktion, som tømmer en treeview.
* Tilføj en linje til hovedprogrammet, der initialiserer GUI'en ved at fylde treeviewen fra databasen.

## Create, update, delete

* Skriv funktioner til oprettelse, opdatering og sletning af poster i databasen. (Tænk også på hard delete vs. soft delete.)
* Udvid GUI'en til at kalde disse funktioner fra GUI'en.
* Forbind GUI-knapperne med denne funktioner.

## Rejser og bookinger

Kopier koden for klassen Kunder (eller hvad du nu kaldte den) to gange og derefter tilpas kopien til henholdsvis klassen Rejser og klassen Bookinger.

## Funktionslag

Før en booking bliver oprettet skal der tjekkes om der kan bookes flere passagerer på bussen uden at overskride dets maksimale transportkapacitet. Skriv en tilsvarende funktion i funktionslaget.

# Gør dit program mere robust

Find et eller to eksempler hvor brugeren kunne gøre noget forkert og ville crashe programmet eller få programmet til at gør noget forkert. Forhindr, at programmet crasher eller dummer sig ved denne fejlbetjening. Du har muligvis brug for en try-except-statement.