This is our database design document, this is what was used during development of the various models and tasks for hitting the milestones.

Here is a link to the git repository:

https://git.gvk.idi.ntnu.no/course/idatg2204/idatg2204-2021-workspace/elvisa/databaseproject

Logical Model

IndividualStores(customer_id, name, start_date, end_date, shipping_address, negotiated_price, franchises_id, can_order_independently, recive_shipments_directly)
PK customer_id
FK franchises_id references Franchises(customer_id)

Franchises(customer_id, name, start_date, end_date, shipping_address, negotiated_price)
PK customer_id

TeamSkiers(customer_id, name, start_date, end-date, birth_date, club, skies_pr_year)
PK customer_id
FK customer_id references customer(customer_id)

CustomerRep(**employee_nr, name, department_affiliation**)
PK employee_nr

StoreKeeper(**employee_nr, name, department_affiliation**)
PK employee_nr

ProductionPlanner(**employee_nr, name, department_affiliation**)
PK employee_nr

Production_plans(plan_name, employee_nr, start_date, end_date)
PK plan_name
FK employee references Employee(employee_nr)

Orders(order_nr, customerId)
PK order_nr
FK customerId references Customer(customer_id)

Skiis(product_id, type_of_skiing, temperature, grip_system, size, weight_class, description, historical, photo_url, msrpp)
PK product_id

Transitions(transition_id, order_nr, state_name, created_by)
PK transition_id
FK order_nr references Orders(order_nr)
FK state_name references State(state_name)
FK created_by references Customer(created_by)

History(transition_id, start_date, end_date)
PK transition_id, start_date
FK transition_id references Transitions(transition_id)

Shipments(shipment_nr, franchise_name, shipping_address, pickup_date, state, driver_id, transporter_name, order_nr)
PK shipment_nr
FK transporter_name references Transporter(company_name)
FK order_nr references Transporter(order_nr)

Transporter(companyName)
PK companyName

ProductionPlansOnSkiis(plan_name, product_id, number_of_skiis, skiis_produced)
FK plan_name references ProductionPlans(plan_name)
FK product_id references Skiss(product_id)

AuthToken(token, permission)
PK token

SQL commands

```
CREATE TABLE customer (
        customer_id int NOT NULL AUTO_INCREMENT,
        name varchar(255),
        start_date DATE,
        end_date DATE,
        PRIMARY KEY (customer_id)
);


CREATE TABLE Franchises (
        customer_id INT NOT NULL,
        shipping_address varchar(255),
        negotiated_price INT,
        PRIMARY KEY(customer_id),
        FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);




CREATE TABLE Individual_stores (
        customer_id INT,
        shipping_address varchar(255),
        negotiated_price INT,
        franchise_id INT,
        PRIMARY KEY (customer_id),
        FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
        FOREIGN KEY (franchise_id) REFERENCES franchises(customer_id)
);


CREATE TABLE Team_skiers (
        customer_id INT NOT NULL,
        birth_date DATE,
        club varchar(255),
        skies_pr_year INT,
        PRIMARY KEY(customer_id),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);

CREATE TABLE Orders (
        order_nr INT NOT NULL AUTO_INCREMENT,
        customer_id INT NOT NULL,
        PRIMARY KEY(order_nr),
        FOREIGN KEY (customer_id) REFERENCES customer(customer_id)
);
```

```
CREATE TABLE State(
        state_name varchar(255) NOT NULL,
        state_value INT NOT NULL
        PRIMARY KEY (state_name)
);

CREATE TABLE Transition(
   transition_id INT NOT NULL AUTO_INCREMENT,
   created_employee_nr INT NOT NULL,
   state_name varchar(255) NOT NULL,
   order_nr INT NOT NULL,
PRIMARY KEY (transition_id),
FOREIGN KEY (created_employee_nr) REFERENCES employee(employee_nr),
FOREIGN KEY (state_name) REFERENCES state(state_name),
FOREIGN KEY (order_nr) REFERENCES Orders(order_nr)
);


CREATE TABLE History(
   transition_id INT NOT NULL,
   start_date DATE NOT NULL,
   end_date DATE,
   PRIMARY KEY (transition_id, start_date),
   FOREIGN KEY (transition_id) REFERENCES Transition(transition_id)
);




CREATE TABLE Skiis(
        product_id INT NOT NULL AUTO_INCREMENT,
        model varchar(255),
        type_of_skiing varchar(255),
        temperature INT,
        grip_system varchar(255),
        size INT,
        weight_class INT,
        description varchar(255),
        historical BIT NOT NULL DEFAULT 0,
        photo_url varchar(255),
        msrpp INT,
        PRIMARY KEY(product_id )
);
```

```
CREATE TABLE SkiOrders(
        order_nr INT NOT NULL,
        ski_id INT NOT NULL,
        skiis_ordered INT,
        skiis_ready INT,
        PRIMARY KEY(order_nr, ski_id),
        FOREIGN KEY order_nr references Orders(order_nr),
        FOREIGN KEY ski_id references Skiis(ski_id)
)



CREATE TABLE Shipments (
   shipment_nr INT NOT NULL AUTO_INCREMENT,
   order_nr INT NOT NULL,
   transporter_name varchar(255),
   franchise_name varchar(255),
   shipping_address varchar(255),
   pickup_date DATE,
   state varchar(255),
   driver_id INT,
   PRIMARY KEY(shipment_nr),
   FOREIGN KEY (order_nr) REFERENCES orders(order_nr),
   FOREIGN KEY (transporter_name) REFERENCES transporter(transporter_name)
);




CREATE TABLE transporter(
transporter_name varchar(255) NOT NULL,
PRIMARY KEY(transporter_name)
);

ALTER TABLE production_plans_on_skiis ENGINE=InnoDB

CREATE TABLE production_plans_on_skiis(
        plan_name varchar(255) NOT NULL,
        product_id INT NOT NULL,
        number_of_skiis INT NOT NULL,
        PRIMARY KEY(plan_name, product_id),
        FOREIGN KEY (plan_name) references production_plans(plan_name),
        FOREIGN KEY (product_id) references skiis(product_id)
)
```

```
CREATE TABLE production_plans(
plan_name varchar(255) UNIQUE ,
responsible_employee_nr INT NOT NULL,
start_date DATE,
end_date DATE.
PRIMARY KEY(plan_name),
FOREIGN KEY (responsible_employee_nr) REFERENCES
production_planner(employee_nr)
)

ALTER TABLE production_plans_on_skiis
ADD FOREIGN KEY (product_id) REFERENCES skiss(product_id);

CREATE TABLE Employee(
        employee_nr INT NOT NULL AUTO_INCREMENT,
        name varchar(255),
        department_affiliation varchar(255),
        PRIMARY KEY (Employee_nr)
);

CREATE TABLE Customer_rep(
        employee_nr INT NOT NULL,
        PRIMARY KEY(employee_nr),
        FOREIGN KEY (employee_nr) REFERENCES Employee(employee_nr)
);

CREATE TABLE Storekeeper(
        employee_nr INT NOT NULL,
        PRIMARY KEY(employee_nr),
        FOREIGN KEY (employee_nr) REFERENCES Employee(employee_nr)
);

CREATE TABLE Production_Planner(
        employee_nr INT NOT NULL,
        PRIMARY KEY(employee_nr),
        FOREIGN KEY (employee_nr) REFERENCES employee(employee_nr)
);


CREATE TABLE SkiOrders(
        order_nr INT NOT NULL AUTO_INCREMENT,
        ski_id INT NOT NULL,
        skiis_ordered INT NOT NULL,
        skiis_ready INT,
        PRIMARY KEY(order_nr, ski_id),
        FOREIGN KEY (order_nr) REFERENCES Orders(order_nr),
        FOREIGN KEY (ski_id ) REFERENCES  Skiis(product_id)
)
```

```
ALTER TABLE SkiOrders ADD
skiis_ordered INT,

CREATE TABLE auth_token(
        token VARCHAR(255) NOT NULL,
        permission VARCHAR(255) NOT NULL,
        PRIMARY KEY(token),
);
```

Insert SQL commands

```
INSERT INTO Skiis (product_id, model, type_of_skiing, temperature, grip_system, size,
weight_class, description, historical, photo_url, msrpp)
VALUES ('1', 'Active', 'skate', 'cold', 'wax', '197','60-70','For active persons.', 1,
'https://i1.adis.ws/i/madshus/madshus_1920_race-speed-skate?w=412&fmt=webp&bg=white&protocol=https&dpi=72', '4500'),
('2', 'Race Pro', 'classic', 'warm', 'IntelliGrip', '192','70-80','for people with style.', 1,
'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSKmqhn2yubXLTfZDr3RQLON7a9NGKHSNu6AQ&usqp=CAU', '5600');



INSERT INTO `Customer` (`customer_id`, `name`, `start_date`, `end_date`)
VALUES (1, 'SkiStore', '2008-02-11',   '2022-01-11'),
(2, 'Sport2', '2009-02-05',   '2025-02-03'),
(3, 'Thea Johaug', '2012-04-09',   '2023-05-08')


INSERT INTO `franchises` (`customer_id`, `shipping_address`, `negotiated_price`)
VALUES (2, 'Norvegen 1 2821 Gjøvik', '50')

INSERT INTO `individual_stores` (`customer_id`, `shipping_address`, `negotiated_price`,
`franchise_id`)
VALUES (1, 'Sorvegen 99 2821 Gjøvik', 50, 2)

INSERT INTO `orders`(`order_nr`, `customer_id`)
VALUES (1,1),
(2,2),
(3,3)

INSERT INTO `state`(`state_name`)
VALUES ('new', 1),
('open', 2),
('skis available', 3),
('ready to be shipped', 4),
('shipped', 5)

INSERT INTO `SkiOrders` (`order_nr`, `ski_id`, `skiis_ordered`, `skiis_ready`)
VALUES (1, 1, 5, 0),
(2, 1, 6, 3),
(2, 2, 6, 4),
(3, 1, 2, 2)
```

```
INSERT INTO `employee`(`employee_nr`, `name`, `department_affiliation`) VALUES
('1','Kjartan Holdsbakk','Administration'),
('2','Gyrde Myredal','Sales'),
('3','Olav Løbåt','Production planner')


INSERT INTO `storekeeper`(`employee_nr`) VALUES ('1');
INSERT INTO `customer_rep`(`employee_nr`) VALUES ('2');
INSERT INTO `production_planner`(`employee_nr`) VALUES ('3');

INSERT INTO `transporter`(`transporter_name`) VALUES ('Bosten'), ('Pring'),
('Post-sør-vest'), ('LHD'), ('SPU');


INSERT INTO `transition`(`transition_id`, `created_employee_nr`, `state_name`, `order_nr`)
VALUES ('1', '2', 'new', '1'),
('2', '2', 'open', '1'),
('3', '2', 'new', '2'),
('4', '2', 'open', '2'),
('5', '2', 'skis available', '2'),
('6', '2', 'new', '3'),
('7', '2', 'open', '3'),
('8', '2', 'skis available', '3'),
('9', '1', 'shipped', '3')


INSERT INTO `production_plans`(`plan_name`, `responsible_employee_nr`) VALUES ('Plan
for the easter','3')

INSERT INTO `shipments`(`shipment_nr`, `order_nr`, `transporter_name`, `franchise_name`,
`shipping_address`, `pickup_date`, `state`, `driver_id`)
VALUES ('1', '3', 'Pring', 'Thea Johaug', 'teknologiveien 22 2815 gjøvik', '2021-06-06', 'ready',
'1')


INSERT INTO `production_plans_on_skiis`(`plan_name`, `product_id`, `number_of_skiis`,
`skiis_produced`) VALUES ('Plan for the easter', '2', '20', '11')
```

```
INSERT INTO `history`(`transition_id`, `start_date`, `end_date`)
VALUES ('1', '2020-05-05', '2020-06-06'),
('1', '2020-06-06', '2020-06-07'),
('2', '2020-08-09', ''),
('2', '2020-09-09', '2020-10-11'),
('2', '2020-10-11', ''),
('3', '2021-01-01', '2021-01-11'),
('3', '2021-01-11', '2021-01-14'),
('3', '2021-01-14', '2021-02-03'),
('3', '2021-02-03', '2021-02-03')
```

Endpoint Description

Endpoint	Method	Description
Orders	GET, POST (X,X)	Get all orders. Media type: application/json
Order	GET, PUT, DELETE (X,X,X)	Get a specific order Media type: application/json
Shipment	POST	Get information about a specific shipment. Media type: application/json
Ski	PUT	Media type: application/json
Skis available	GET (X)	Add information about new skis. Retrieve orders in ski available state. Change the state of skis ready to ship. Media type: application/json
production plan	POST	Uploading production plan. Media type: application/json
customer orders	GET	Get all customer orders Media type: application/json
Split order	POST	Media type: application/json
Four production plan	GET	Media type: application/json
Orders ready	GET (X)	retrieve information about orders ready for shipment, Change the state of the shipment. Media type: application/json
Shipment	PUT	Media type: application/json
Public	GET	Get information about skis. Can filter on model name. Media type: application/json

Endpoint	URL	Description
Orders	http://localhost/orders	Use this to get information about all orders
Order	http://localhost/orders/{id}	Use this to get information about a specific order
Shipment	http://localhost/orders/{id}/shipment	Use this to get shipment information about a specific orders
Ski	http://localhost/skiis/{id}	Gives back information about a ski
Skis available	http://localhost/orders/skis_available	Use this information to get all orders that has skiis available.
Production plan	http://localhost/production_plan	Use this information to upload production
Customer orders	http://localhost/customer/{id}/orders	Get list of orders a customer has made.
Split order	http://localhost/orders/{id}/split	Request for order to be split.
Four production plan	http://localhost/ production-plan/{from-date}-{to-date}	Get a plan summary showing total of skis in timeframe.
Orders ready	http://localhost/orders/ready	Get information about orders ready for shipment.
Shipment	http://localhost/shipment/{id}	Change the state of the shipment when it has been picked up.
Public	http://localhost/public/skis/	Get information about skis.

Resource	Representation
Orders	[order*]
Order	{“order_nr”: “integer”, “state”: “string”, “franchise_name”: “string”, “shipping_address”: “string”, “shipment_state”: “string”,“skiis”: [“skiis_id”: “integer”, “skiis_ordered”: “integer”, “skiis_ready”: “integer”]}
Shipment	{"shipment_nr": "integer", "order_nr": "integer", "transporter_name": "string", "franchise_name": "string", "shipping_adress": "string", "pickup_date": "string", "state": "string", "driver_id": "integer"}
Ski	{"product_id": "integer", "type_of_skiing": "string", "temperature": "string", "grip_system": "string", "size": "string", "weight_class": "string", "description": "string", "historical": "boolean", "photo_url": "string", "msrpp": "string"}
Skiis available	[Ski*]
Production_plans	[production_plan*]
production_plan	{"plan_name": "string", "responsible_employee": "integer", “skiis”: [skiis_id”: “integer”, “skiis_ordered”: “integer”]}
Customer orders	[customer order*]
Customer order	Order*
Split orders	[split order*]
Split order	customer order*
Four production plans	[four production plan*]
Four production plan	
Orders ready	[Order ready*]
Orders ready	
Shipments	[Shipment*]
Shipment	
Public	[Ski*]

Milestone-2 API & UNIT TESTS

In regards to whom will be developing we won't separate that responsibility harshly. In reality we might end up with a skew of 60-40 but we both will be writing tests. We believe that it's something that we both should do in the project.

The tests will be developed as the functions in our program are written. First you write minimal code to get the function to work and then you write the according test for that function. We then expand the test more broadly and then change the function based on that. Doing test-driven coding (which would be during code development) is something we are familiar with and will employ in this project.

We plan on implementing these following tests:

1. For all API tests we will check that the header is json format. As everything should be json.
2. We will also attempt to write the more obvious ones such as doing GET on an existing resource and GET on a non-existent resource.
3. We write a test which will test this endpoint: http://localhost/skiis/{id}. This test will check for a 200 response form the server and it will contain the data that is expected. When the test gets the data it compares and it returns a passed test if the data matches.
4. For most of the endpoints regarding a specific ID, we will implement a test to check for non-existing entries. This test is expected to return a 404, and this is what will be it's pass criteria.
5. For the endpoint: http://localhost/orders/{id}/split we will write a test such that you should be able to check whether an order was actually split. This test will also check if the data was correct as a safety measure.
6. For the endpoint: http://localhost/orders/orders_ready we will write a test that should check if the correct data is returned, and that all orders return have state "ready to be shipped"
7. http://localhost/orders/{id}/shipment endpoint will be tested for the specific information. We have the database so we can have that data as well to check with beforehand.
8. For endpoint http://localhost/production_plan we will write a post-request test.

We will check if the post was successful and that the number of rows in the table increased from what it was.

9. We will check that authorized requests (with valid token) to the endpoints will work, except for public/skis. We will also test that unauthorized requests to the endpoints will not work.
10. **UNIT TESTS**: For most of the endpoints we will define unit tests that will check whether something is an invalid resource. For skis for example for testing skis we would use “self-assertEquals” with “isValidEndpoint”.
11. For this endpoint, http://localhost/orders we will check the header content and that for a success of 200.
12. For asserting counts in our post tests we will use “assertCount” on the amount of fields currently but with plus one because we just did a post request.