Requirements specification

NOTE: the parameters in {} are optional

1. Change **/api/stands/all** to just **/api/stands{?map=true}**
   1. Stand
   2. City
   3. Township
   4. Status
2. **/api/stands/reservations{?map=true}**
   1. Stand
   2. City
   3. Township
   4. Status (**no need for this as this is implied in the name of the api function**)
   5. Client name
   6. Reservation date
   7. Expiry date (add status column in reservations table)
   8. SELECT (DATE '2001-02-16', INTERVAL '7 days') OVERLAPS

(DATE '2001-10-30', DATE '2002-10-30');

* 1. SELECT (ReservationDATE '2001-02-16', INTERVAL '7 days') OVERLAPS

(reservationDATE '2001-10-30', currentDATE '2002-10-30' );

SELECT (DATE '2017-02-21', INTERVAL '7 days') OVERLAPS

(DATE '2001-10-30', LOCALTIMESTAMP);

1. **/api/stands/available{?map=true}**
   1. Stand
   2. City
   3. Township
   4. Status

// code to return map

SELECT 'FeatureCollection' As type, array\_to\_json(array\_agg(f)) As features FROM (SELECT 'Feature' As type, ST\_AsGeoJSON(cadastre.geom)::json As geometry, row\_to\_json ((SELECT l FROM (SELECT cadastre.standid AS standid, cities.name AS city, townships.name AS township) AS l)) AS properties FROM cadastre, cities, townships WHERE NOT EXISTS (SELECT \* FROM reservations r WHERE r.standid = cadastre.standid AND (reservationdate+period\*interval '0 day', reservationdate+period\*interval '1 day') OVERLAPS (reservationdate+period\*interval '1 day', LOCALTIMESTAMP)) AND NOT EXISTS (select null from soldstands where soldstands.standid = cadastre.standid) AND cadastre.cityid = cities.cityid AND cadastre.townshipid = townships.townshipid ORDER BY cadastre.standid, townships.name, cities.name) As f

// no map returned by the following code

SELECT \*

FROM cadastre c

WHERE NOT EXISTS (

SELECT \*

FROM reservations r WHERE r.standid = c.standid AND (reservationdate+period\*interval '0 day', reservationdate+period\*interval '1 day') OVERLAPS (reservationdate+period\*interval '1 day', LOCALTIMESTAMP)

)

AND NOT EXISTS (select null from soldstands where soldstands.standid = c.standid)

ORDER BY c.standid

1. **/api/stands/sold{?map=true}**
   1. Stand
   2. City
   3. Township
   4. Status
   5. Client name
   6. Sales date
   7. Payment to date
   8. Outstanding balance
   9. Price

SELECT cadastre.standid AS standid, cities.name AS city, townships.name AS township FROM cadastre, cities, townships, soldstands WHERE cadastre.cityid = cities.cityid AND cadastre.townshipid = townships.townshipid

INNER JOIN soldstands ON cadastre.standid = soldstands.standid

1. **/api/stands{?map=true}/:id**
   1. Stand
   2. City
   3. Township
   4. Status
   5. Timeline URL
2. /api/timeline/stand/:id
3. /api/timeline/client/:id

SELECT 'FeatureCollection' AS type, array\_to\_json(array\_agg(f)) AS features FROM (SELECT 'Feature' AS type, ST\_AsGeoJSON(cadastre.geom, 6)::json As geometry, row\_to\_json((SELECT l FROM (SELECT wateraccount.townshipid, wateraccount.dsg\_num, meterconnection.meterid, meterconnection.accountnumber, meterconnection.connectiontype, accountholder.surname) AS l)) AS properties FROM wateraccount INNER JOIN cadastre ON wateraccount.townshipid = cadastre.townshipid AND wateraccount.dsg\_num = cadastre.dsg\_num INNER JOIN meterconnection ON wateraccount.accountnumber = meterconnection.accountnumber INNER JOIN accountholder ON meterconnection.accountnumber = accountholder.accountnumber GROUP BY wateraccount.townshipid, wateraccount.dsg\_num, meterconnection.meterid, meterconnection.accountnumber, meterconnection.connectiontype, accountholder.surname, cadastre.geom) AS f

SELECT cadastre.standid AS standid, cities.name AS city, townships.name AS township, reservations.reservationdate AS reservationdate, reservations.reservationdate+period\*INTERVAL'1 day' AS expirydate FROM cadastre, cities, townships, reservations WHERE cadastre.standid IN (SELECT standid FROM reservations WHERE (reservationdate+period\*interval '0 day', reservationdate+period\*interval '1 day') OVERLAPS (reservationdate+period\*interval '1 day', LOCALTIMESTAMP)) AND cadastre.standid = reservations.standid AND cadastre.cityid = cities.cityid AND cadastre.townshipid = townships.townshipid

SELECT 'FeatureCollection' As type, array\_to\_json(array\_agg(f)) As features FROM (SELECT 'Feature' As type, ST\_AsGeoJSON(cadastre.geom)::json As geometry, row\_to\_json ((SELECT l FROM (SELECT cadastre.standid AS standid, cities.name AS city, townships.name AS township, reservations.reservationdate AS reservationdate, reservations.reservationdate+period\*INTERVAL'1 day' AS expirydate) AS l)) AS properties FROM cadastre, cities, townships, reservations WHERE cadastre.standid IN (SELECT standid FROM reservations WHERE (reservationdate+period\*interval '0 day', reservationdate+period\*interval '1 day') OVERLAPS (reservationdate+period\*interval '1 day', LOCALTIMESTAMP)) AND cadastre.standid = reservations.standid AND cadastre.cityid = cities.cityid AND cadastre.townshipid = townships.townshipid) As f

Payments

SELECT receipts.standid, sum(receipts.amount) totalpayment, soldstands.price price, cities.name city\_name, townships.name township\_name, soldstands.clientid clientid, soldstands.price price, receipts.amount amount, clients.name firstname, clients.surname surname, clients.email email FROM receipts,soldstands, cities, townships, clients

INNER JOIN soldstands ON receipts.standid = soldstands.standid

GROUP BY receipts.standid, soldstands.price, cities.name, townships.name, soldstands.clientid, soldstands.price, receipts.amount, clients.name, clients.surname, clients.email