```yaml
# -------------------------

service: taxis-handler
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: /registerTaxi
  script: registerTaxi.app

- url: /startTrip
  script: startTrip.app

- url: /endTrip
  script: endTrip.app

- url: /checkFacturacion
  script: checkFacturacion.app

libraries:

- name: webapp2
  version: latest

- name: jinja2
  version: latest
```

```python

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime

from google.appengine.ext import ndb
from Models import *

import webapp2
from webapp2_extras import json

# [START startTrip]
class StartTrip(webapp2.RequestHandler):

  def get(self):

    driver_id = self.request.get("driverId")
    driver_key = ndb.Key(urlsafe=driver_id)
    new_id = ndb.Model.allocate_ids(size=1, parent=driver_key)[0]
    trip_key = ndb.Key('Trip', new_id, parent=driver_key)

    pickup_datetime = self.request.get("datetime")
    pickup_latitude = self.request.get("latitude")
    pickup_longitude = self.request.get("longitude")

    trip = Trip(key=trip_key,
      pickup_datetime=datetime.strptime(pickup_datetime, "%Y-%m-%d %H:%M:%S"),
      pickup_location=ndb.GeoPt(pickup_latitude, pickup_longitude))

    trip.put();
    self.response.content_type = 'application/json'
    response_json = {
      'success': 'OK',
      'tripKey': trip_key.urlsafe()
    }
    self.response.write(json.encode(response_json))

# [START app]
app = webapp2.WSGIApplication([
  ('/startTrip', StartTrip)
], debug=True)
# [END app]
```

Apr 24, 18 17:43 **registerTaxi.py** Page 1/1

```

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime

from google.appengine.ext import ndb
from Models import *

#import jinja2
import webapp2
from webapp2_extras import json

# [START registerTaxi]
class RegisterTaxi(webapp2.RequestHandler):

  def get(self):

    driver_name = self.request.get("driverName")

    taxi = Taxi(driver_name=driver_name)

    taxi_key = taxi.put()
    taxi_key_urlsafe = taxi_key.urlsafe()

    self.response.content_type = 'application/json'
    response_json = {
      'success': 'OK',
      'taxiKey': taxi_key_urlsafe
    }
    self.response.write(json.encode(response_json))

# [START app]
app = webapp2.WSGIApplication([
  ('/registerTaxi', RegisterTaxi)
], debug=True)
# [END app]
```

Apr 23, 18 17:10 **queue.yaml** Page 1/1

```
queue:
- name: trips-per-day
  mode: pull
```

Apr 24, 18 22:10 **Models.py** Page 1/1

```python
from google.appengine.ext import ndb

# [START taxi]
class Taxi(ndb.Model):
  """A main model for representing a taxi driver."""
  driver_name = ndb.StringProperty(indexed=True);
# [END taxi]

# [START trip]
class Trip(ndb.Model):
  """A main model for representing a taxi trip."""
  pickup_datetime = ndb.DateTimeProperty(indexed=True)
  pickup_location = ndb.GeoPtProperty(indexed=False)
  dropoff_datetime = ndb.DateTimeProperty(indexed=True)
  dropoff_location = ndb.GeoPtProperty(indexed=False)
  distance = ndb.FloatProperty(indexed=False)
# [END trip]

# [START statistic]
class DailyStatistic(ndb.Model):
  """A main model for representing a daily trips statistic."""
  date = ndb.DateProperty(indexed=True)
  trips = ndb.IntegerProperty(indexed=False)
# [END statistic]
```

Apr 23, 18 3:13 **index.yaml** Page 1/1

```yaml
indexes:

# AUTOGENERATED

# This index.yaml is automatically updated whenever the dev_appserver
# detects that a new type of query is run.  If you want to manage the
# index.yaml file manually, remove the above marker line (the line
# saying "# AUTOGENERATED").  If you want to manage some indexes
# manually, move them above the marker line.  The index.yaml file is
# automatically uploaded to the admin console when you next deploy
# your application using appcfg.py.

- kind: Trip
  ancestor: yes
  properties:
  - name: pickup_datetime
    direction: desc
```

```python

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime
import logging

from google.appengine.ext import ndb
from google.appengine.api import memcache
from google.appengine.api import datastore
from google.appengine.api import taskqueue
from Models import *
from Configuration import *

import webapp2
from webapp2_extras import json

# [START endTrip]
class EndTrip(webapp2.RequestHandler):

  def get(self):

    trip_id = self.request.get("tripId")
    trip_key = ndb.Key(urlsafe=trip_id)

    trip = memcache.get(trip_key.urlsafe())
    if trip is None:
      trip = trip_key.get()
      if trip is None:
        logging.error("Received endTrip request for invalid trip id: %s", trip_id)
        return
      memcache.add(trip_key.urlsafe(), trip)

    dropoff_datetime = self.request.get("datetime")
    dropoff_latitude = self.request.get("latitude")
    dropoff_longitude = self.request.get("longitude")
    distance = float(self.request.get("distance"))

    trip.dropoff_datetime = datetime.strptime(dropoff_datetime, "%Y-%m-%d %H:%
M:%S")
    trip.dropoff_location = ndb.GeoPt(dropoff_latitude, dropoff_longitude)
    trip.distance = distance

    trip.put();

    # pass to pull queue for dayli trips counter
    tag = trip.pickup_datetime.strftime("%Y-%m-%d")
    q = taskqueue.Queue(Configuration.DailyTripsTaskQueue)
    tasks = []
    payload_str = '1'
    tasks.append(taskqueue.Task(payload=payload_str, method='PULL', tag=tag))
    q.add(tasks)

    self.response.content_type = 'application/json'
    response_json = {
      'success': 'OK'
    }
    self.response.write(json.encode(response_json))

# [START app]
app = webapp2.WSGIApplication([
  ('/endTrip', EndTrip)
], debug=True)
# [END app]
```

```yaml
dispatch:

- url: "*/startTrip"
  service: taxis-handler

- url: "*/registerTaxi"
  service: taxis-handler

- url: "*/endTrip"
  service: taxis-handler

- url: "*/checkFacturacion"
  service: taxis-handler

- url: "*/tasks/daily-trip-summary"
  service: daily-trip-summary

- url: "*/adminQuery"
  service: admin-query

- url: "*/dailyTripsStatistic"
  service: daily-trips-statistic
```

```python

# [START imports]
import os
import urllib
from urlparse import *

import jinja2
import webapp2
from webapp2_extras import json

JINJA_ENVIRONMENT = jinja2.Environment(
  loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
  extensions=['jinja2.ext.autoescape'],
  autoescape=True)
# [END imports]

# [START main]
class MainPage(webapp2.RequestHandler):

  def get(self):

    template = JINJA_ENVIRONMENT.get_template('default-index.html')
    self.response.write(template.render())

# [END main]

# [START app]
app = webapp2.WSGIApplication([
  ('/', MainPage)
], debug=True)
# [END app]
```

Whoops

You are not supposed to be here! {% endautoescape %}

Apr 23, 18 17:41 **daily−trip−summary.yaml** Page 1/1

```yaml
# ------------------------

service: daily-trip-summary
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: /tasks/daily-trip-summary
  script: dailyTripSummary.app

libraries:
- name: webapp2
  version: latest
```

Apr 26, 18 16:30 **dailyTripSummary.py** Page 1/1

```python

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime
from Configuration import *

from google.appengine.ext import ndb
from google.appengine.api import memcache
from google.appengine.api import datastore
from google.appengine.api import taskqueue
from Models import *

import webapp2
import logging

# [START endTrip]
class DailyTripSummary(webapp2.RequestHandler):

  def get(self):

    q = taskqueue.Queue(Configuration.DailyTripsTaskQueue)
    tasks = q.lease_tasks_by_tag(Configuration.TasksLeaseDurationSeconds, Config
uration.AmountOfTasksPerLease)
    logging.info(tasks)

    numTrips = len(tasks)
    if numTrips ≤ 0:
      return

    tag = tasks[0].tag
    statistic_key = ndb.Key('DailyStatistic', tag)
    statistic = memcache.get(tag)
    if statistic is None:
      statistic = statistic_key.get()
      if statistic is None:
        statistic = DailyStatistic(key=statistic_key, date=datetime.strptime(tag
, "%Y-%m-%d"), trips=0)
      memcache.add(tag, statistic)
    statistic.trips += numTrips
    statistic.put()

    q.delete_tasks(tasks)

# [START app]
app = webapp2.WSGIApplication([
  ('/tasks/daily-trip-summary', DailyTripSummary)
], debug=True)
# [END app]
```

```yaml
# ------------------------

service: daily-trips-statistic
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: /dailyTripsStatistic
  script: dailyTripsStatistic.app

libraries:
- name: webapp2
  version: latest

- name: jinja2
  version: latest
```

```python

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime
from datetime import timedelta

from google.appengine.ext import ndb
from google.appengine.api import memcache
from Models import *

import jinja2
import webapp2
from webapp2_extras import json

JINJA_ENVIRONMENT = jinja2.Environment(
  loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
  extensions=['jinja2.ext.autoescape'],
  autoescape=True)
# [END imports]

# [START checkFacturacion]
class DailyTripsStatistic(webapp2.RequestHandler):

  def get(self):

    template_values = {
            'date': datetime.today().strftime('%Y-%m-%d')
        }

    template = JINJA_ENVIRONMENT.get_template('daily-trips-statistic-index.html')
    self.response.write(template.render(template_values))

  def post(self):

    date = self.request.get("date")
    statistic_key = ndb.Key('DailyStatistic', date)

    statistic = memcache.get(statistic_key.urlsafe())
    if statistic is None:
      statistic = statistic_key.get()
      if statistic is None:
        num_trips = 0
      else:
        memcache.add(statistic_key.urlsafe(), statistic)
        num_trips = statistic.trips
    else:
      num_trips = statistic.trips

    showTrips = True
    template_values = {
      'date': date,
      'number_of_trips': num_trips,
      'showTrips': showTrips
    }

    template = JINJA_ENVIRONMENT.get_template('daily-trips-statistic-index.html')
    self.response.write(template.render(template_values))

# [END checkFacturacion]

# [START app]
app = webapp2.WSGIApplication([
  ('/dailyTripsStatistic', DailyTripsStatistic)
], debug=True)
# [END app]
```

Viajes totales por dia

{% if showTrips %} Number of trips: {{number_of_trips}} {% endif %} {% endautoescape %}

```
cron:
- description: "daily trips summary"
  url: /tasks/daily-trip-summary
  target: daily-trip-summary
  schedule: every 8 hours
```

```

class Configuration():
  AmountOfTasksPerLease = 100
  TasksLeaseDurationSeconds = 300

  AdminQueryDefaultTripsPerPage = 10

  DailyTripsTaskQueue = "trips-per-day"
```

```

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime
from datetime import timedelta

from google.appengine.ext import ndb
from Models import *

import jinja2
import webapp2
from webapp2_extras import json

JINJA_ENVIRONMENT = jinja2.Environment(
  loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
  extensions=['jinja2.ext.autoescape'],
  autoescape=True)
# [END imports]

# [START checkFacturacion]
class CheckFacturacion(webapp2.RequestHandler):

  def get(self):

    template_values = {
            'facturacion_results': {}
        }

    template = JINJA_ENVIRONMENT.get_template('check-facturacion-index.html')
    self.response.write(template.render(template_values))

  def post(self):

    driver_id = self.request.get("driverId")
    driver_key = ndb.Key(urlsafe=driver_id)

    query_date = datetime.strptime(self.request.get("query_date"), "%Y-%m-%d")

    facturacion_query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date, Tr
ip.pickup_datetime < query_date + timedelta(days=1)), ancestor=driver_key).order
(-Trip.pickup_datetime)
    facturacion_results = facturacion_query.fetch()

    template_values = {
      'facturacion_results': facturacion_results
    }

    template = JINJA_ENVIRONMENT.get_template('check-facturacion-index.html')
    self.response.write(template.render(template_values))

# [END checkFacturacion]

# [START app]
app = webapp2.WSGIApplication([
  ('/checkFacturacion', CheckFacturacion)
], debug=True)
# [END app]
```

Resumen de FacturaciÃ³n

{% for trip in facturacion_results %}
Pickup time: {{trip.pickup_datetime}}, Pickup location: {{trip.pickup_location}}, Dropoff time: {{trip.dropoff_datetime}}, Dropoff location: {{trip.dropoff_location}}, Distance: {{trip.distance}}
{% endfor %}
{% endautoescape %}

Apr 24, 18 22:49 **app.yaml** Page 1/1

```
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: "/"
  script: default.app

libraries:
- name: webapp2
  version: latest

- name: jinja2
  version: latest
```

Apr 24, 18 17:59 **admin–query.yaml** Page 1/1

```
# -----------------------

service: admin-query
runtime: python27
api_version: 1
threadsafe: true

handlers:

- url: /adminQuery
  script: adminQuery.app

libraries:
- name: webapp2
  version: latest

- name: jinja2
  version: latest
```

```python

# [START imports]
import os
import urllib
from urlparse import *
from datetime import datetime
from datetime import timedelta

from google.appengine.datastore.datastore_query import Cursor
from google.appengine.ext import ndb
from google.appengine.api import memcache
from Models import *
from Configuration import *

import jinja2
import webapp2
from webapp2_extras import json
import logging

JINJA_ENVIRONMENT = jinja2.Environment(
  loader=jinja2.FileSystemLoader(os.path.dirname(__file__)),
  extensions=['jinja2.ext.autoescape'],
  autoescape=True)
# [END imports]

# [START adminQuery]
class AdminQuery(webapp2.RequestHandler):

  def get(self):

    template_values = {
            'results': {},
      'query_date_from': "2018-04-01",
      'query_date_to': datetime.today().strftime('%Y-%m-%d'),
      'driverId': "",
      'prev_cursor': "",
      'next_cursor': "",
      'TRIPS_PER_PAGE': Configuration.AdminQueryDefaultTripsPerPage
        }

    template = JINJA_ENVIRONMENT.get_template('admin-query-index.html')
    self.response.write(template.render(template_values))

  def post(self):

    TRIPS_PER_PAGE = int(self.request.get("TRIPS_PER_PAGE"))

    # IF prev BUTTON WAS PRESSED
    is_prev = self.request.get("prev", "") ≠ ""

    # DRIVER QUERY
    driverId = self.request.get("driverId", "")

    # DATE FROM QUERY
    query_date_from = datetime.strptime(self.request.get("query_date_from"), "%Y-%
m-%d")

    # DATE TO QUERY
    query_date_to = datetime.strptime(self.request.get("query_date_to"), "%Y-%m-%d
")

    # CURSOR
    prev_cursor_before = self.request.get('prev_cursor',default_value="")
    next_cursor_before = self.request.get('next_cursor',default_value="")

    # if a driver_id was input, use it to filter
    if driverId ≡ "":
      query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date_from, Trip.pi
ckup_datetime < query_date_to + timedelta(days=1)))
    else:
      driver_key = ndb.Key(urlsafe=driverId).get() # usar memcache
      query = Trip.query(ndb.AND(Trip.pickup_datetime ≥ query_date_from, Trip.pi
ckup_datetime < query_date_to + timedelta(days=1)), ancestor=driver_key)
```

```python

    # --------- PAGE MANAGEMENT
    query_forward = query.order(Trip.pickup_datetime)
    query_reverse = query.order(-Trip.pickup_datetime)

    if is_prev:
      qry = query_reverse
      cursor = ndb.Cursor(urlsafe=prev_cursor_before).reversed() if prev_cursor_
before ≠ "" else None
    else:
      qry = query_forward
      cursor = ndb.Cursor(urlsafe=next_cursor_before) if next_cursor_before ≠ ""
 else None

    # trips in page
    query_results, cursor, more = qry.fetch_page(TRIPS_PER_PAGE, start_cursor=cu
rsor)
    # get driver info for each trip
    results = []
    drivers = {}
    for result in query_results:
      taxiKey = result.key.parent()
      if taxiKey.urlsafe() ¬ in drivers:
        taxi = memcache.get(taxiKey.urlsafe())
        if taxi is None:
          taxi = taxiKey.get()
          if taxi is None:
            logging.error("Received trip exists for taxiKey: %s, but taxi doesn't exist", trip_id)
            taxi = { "driver_name": "MISSING" }
          else:
            memcache.add(taxiKey.urlsafe(), taxi)
        drivers[taxiKey.urlsafe()] = taxi.driver_name
      total_result = { "driver": drivers[taxiKey.urlsafe()], "trip": result }
      results.append(total_result)

    if is_prev:
      prev_cursor_url = cursor.reversed().urlsafe() if more else ""
      next_cursor_url = prev_cursor_before
    else:
      prev_cursor_url = next_cursor_before
      next_cursor_url = cursor.urlsafe() if more else ""
    # ---------

    template_values = {
      'results': results,
      'query_date_from': query_date_from,
      'query_date_to': query_date_to,
      'driverId': driverId,
      'prev_cursor': prev_cursor_url,
      'next_cursor': next_cursor_url,
      'TRIPS_PER_PAGE': TRIPS_PER_PAGE
    }

    template = JINJA_ENVIRONMENT.get_template('admin-query-index.html')
    self.response.write(template.render(template_values))

# [END adminQuery]

# [START app]
app = webapp2.WSGIApplication([
  ('/adminQuery', AdminQuery)
], debug=True)
# [END app]
```

Acceso de administrador

trips per page:

{% for result in results %}
Driver name: {{result.driver}}, Pickup time: {{result.trip.pickup_datetime}}, Pickup location: {{result.trip.pickup_location}}, Dropoff time: {{result.trip.dropoff_datetime}}, Dropoff location: {{result.trip.dropoff_location}}, Distance: {{result.trip.distance}}
{% endfor %}

{% if prev_cursor != "" %} {% endif %} {% if next_cursor != "" %} {% endif %}

{% endautoescape %}

Table of Contents