GasBuddy: Moving an app to the cloud

PRAIRIE DEV CON

CLOUD | MOBILE | WEB | DEV

UnionWare®

bold

D2L DESIRE2LEARN

SOLVERA crafting results

imaginet

Microsoft

online business systems

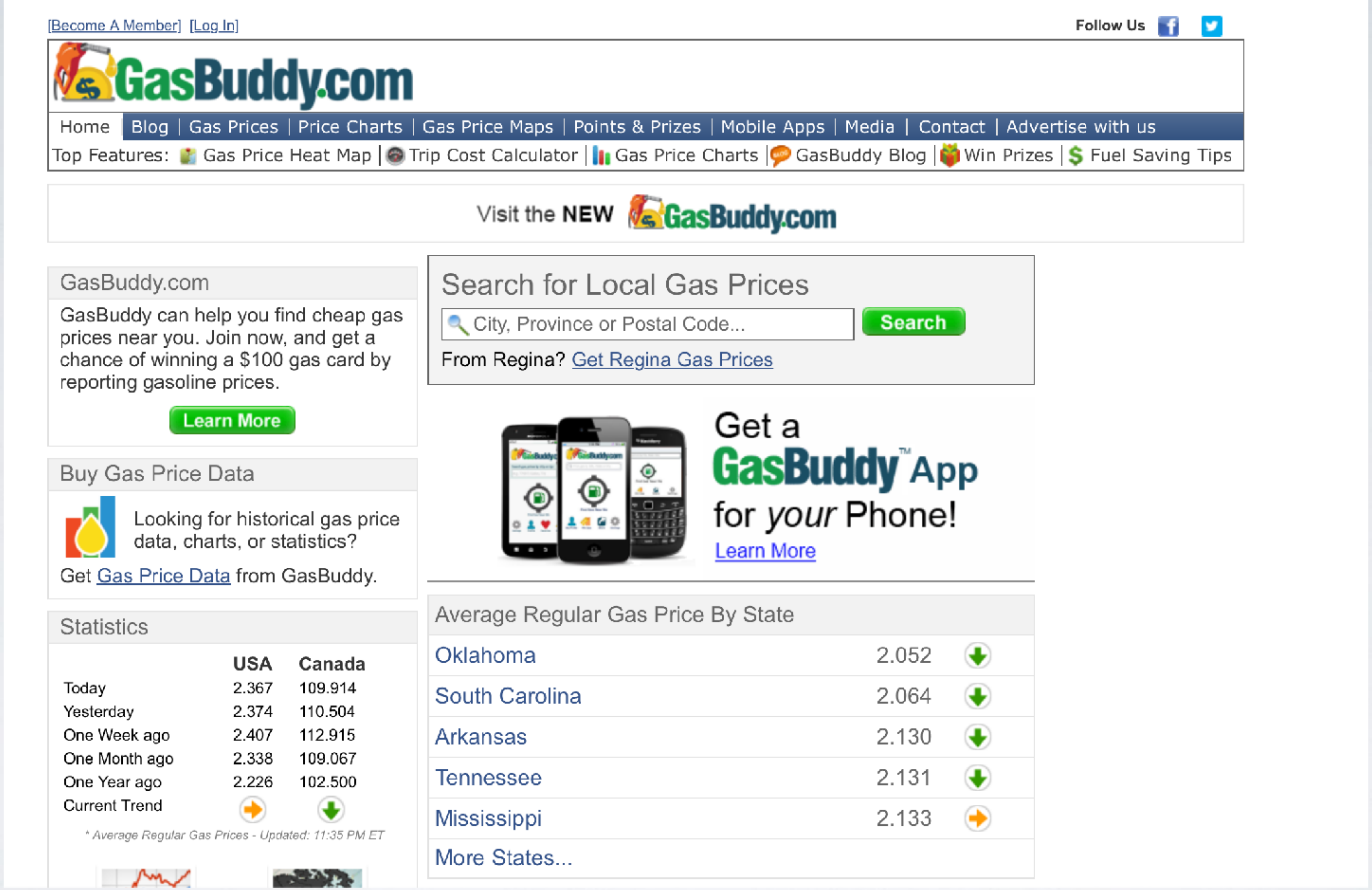
DMT

OUTLINE

- Why move in the first place?
- Refactor case study - Price Reporting
- Rebuild case study - Challenges
- Remove - Analytics

HISTORY OF GASBUDDY

- Started in 2000
- Local Sites reginagasprices.com
- .net + Sql Server



[Become A Member] [Log In]

Follow Us

GasBuddy.com

Home | Blog | Gas Prices | Price Charts | Gas Price Maps | Points & Prizes | Mobile Apps | Media | Contact | Advertise with us

Top Features: Gas Price Heat Map | Trip Cost Calculator | Gas Price Charts | GasBuddy Blog | Win Prizes | Fuel Saving Tips

Visit the NEW GasBuddy.com

GasBuddy.com

GasBuddy can help you find cheap gas prices near you. Join now, and get a chance of winning a $100 gas card by reporting gasoline prices.

Learn More

Buy Gas Price Data

Looking for historical gas price data, charts, or statistics?

Get Gas Price Data from GasBuddy.

Statistics

	USA	Canada
Today	2.367	109.914
Yesterday	2.374	110.504
One Week ago	2.407	112.915
One Month ago	2.338	109.067
One Year ago	2.226	102.500
Current Trend	→	↓

* Average Regular Gas Prices - Updated: 11:35 PM ET

Search for Local Gas Prices

City, Province or Postal Code...

Search

From Regina? Get Regina Gas Prices

Get a GasBuddy™ App for *your* Phone!

Learn More

Average Regular Gas Price By State

Oklahoma	2.052	↓
South Carolina	2.064	↓
Arkansas	2.130	↓
Tennessee	2.131	↓
Mississippi	2.133	→

More States...

GASBUDDY APP

- Mobile app has had 60 million downloads
- 1.5 million price reports per day
- Millions of monthly active users

SCALE UP

- 16 - 2.40 Ghz Intel Xeon Dual Core CPUs
- 1 TB Fusion I/O card (tempdb)
- 512 GB RAM
- High speed disks
- Workhorse is a large SQL server

PROBLEM

- When one machine goes down the app goes down
- In November 2015 a corrupted index brought the app down for 12 hours

GOALS

- Reduce dependancy on one machine
- Stop managing hardware
- Micro-services

OBSTACLES

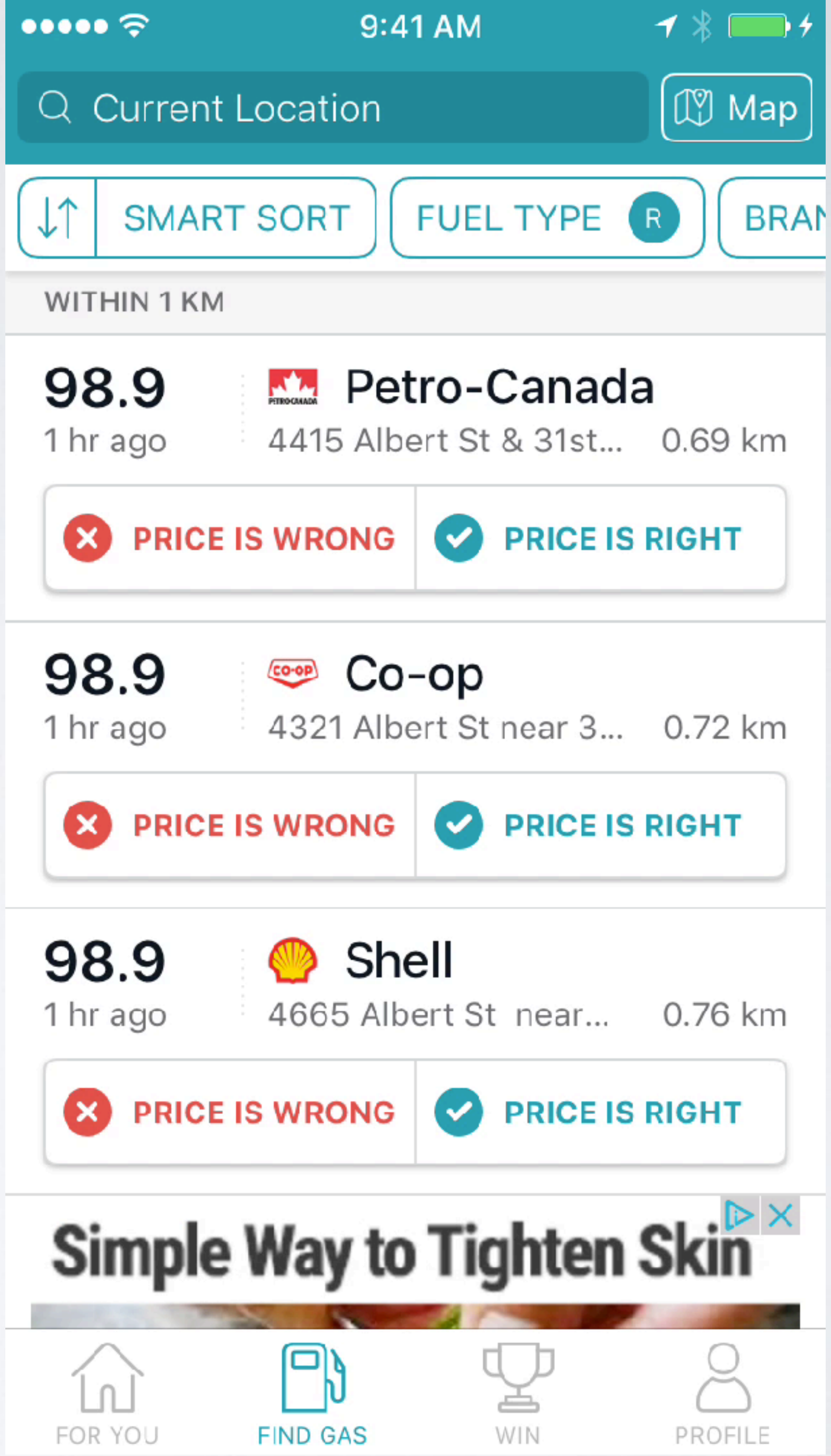
- A BIG relational database
- Complicated Stored Procedures
- High load

REDUCING LOAD

- Refactor - Breakup large stored procedures
- Rebuild - Rebuild systems with data stores in the cloud
- Remove - Get rid of systems if possible move reporting off main db

PRICE REPORTING

Spot a price a station



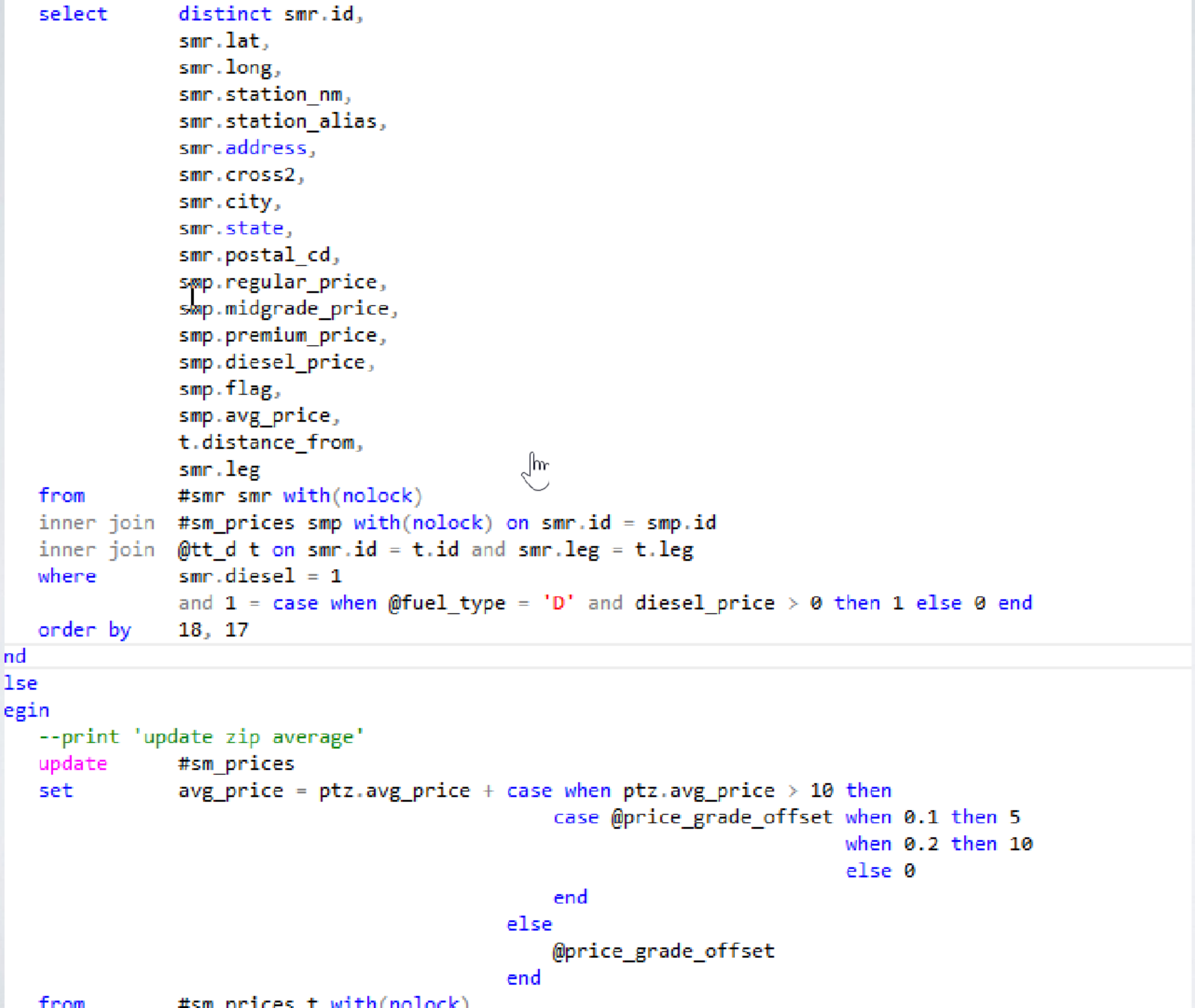
WITHIN 1 KM

98.9 1 hr ago Petro-Canada 4415 Albert St & 31st... 0.69 km
PRICE IS WRONG PRICE IS RIGHT

98.9 1 hr ago Co-op 4321 Albert St near 3... 0.72 km
PRICE IS WRONG PRICE IS RIGHT

98.9 1 hr ago Shell 4665 Albert St near... 0.76 km
PRICE IS WRONG PRICE IS RIGHT

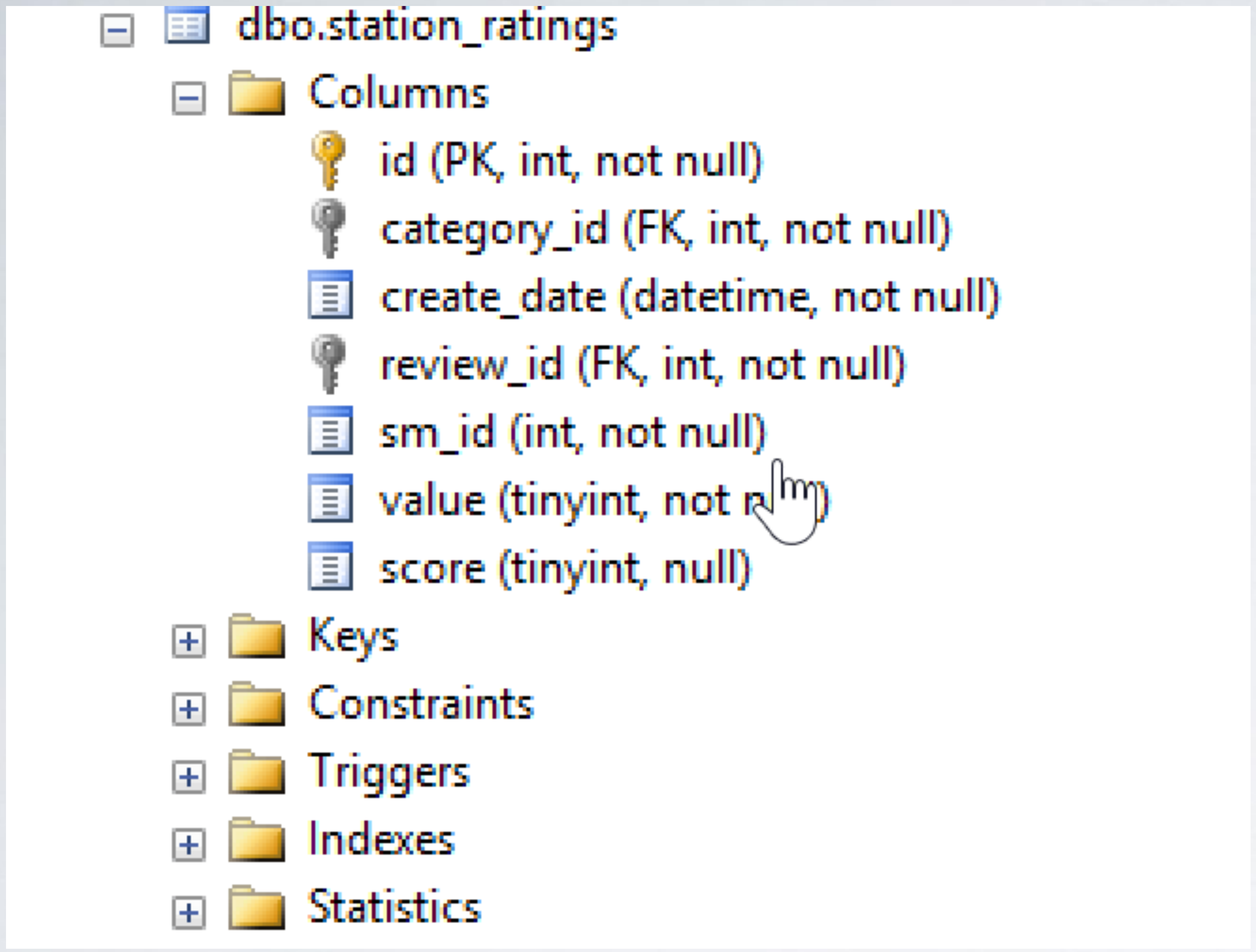
Simple Way to Tighten Skin



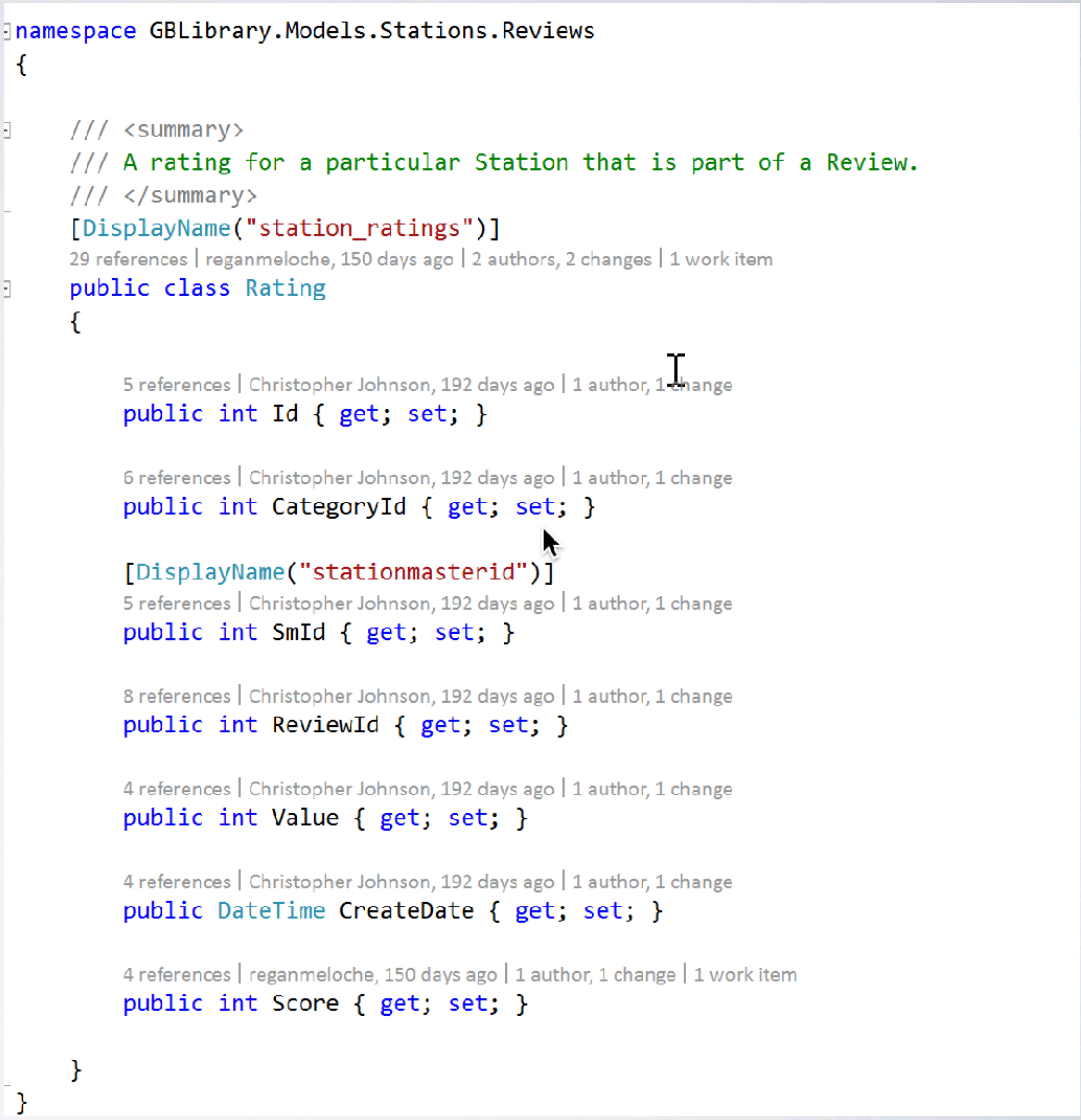
```
   select      distinct smr.id,
               smr.lat,
               smr.long,
               smr.station_nm,
               smr.station_alias,
               smr.address,
               smr.cross2,
               smr.city,
               smr.state,
               smr.postal_cd,
               smp.regular_price,
               smp.midgrade_price,
               smp.premium_price,
               smp.diesel_price,
               smp.flag,
               smp.avg_price,
               t.distance_from,
               smr.leg
   from        #smr smr with(nolock)
   inner join  #sm_prices smp with(nolock) on smr.id = smp.id
   inner join  @tt_d t on smr.id = t.id and smr.leg = t.leg
   where       smr.diesel = 1
               and 1 = case when @fuel_type = 'D' and diesel_price > 0 then 1 else 0 end
   order by    18, 17
nd
lse
egin
   --print 'update zip average'
   update      #sm_prices
   set         avg_price = ptz.avg_price + case when ptz.avg_price > 10 then
                                               case @price_grade_offset when 0.1 then 5
                                                                         when 0.2 then 10
                                                                         else 0
                                               end
                                           else
                                               @price_grade_offset
                                           end
   from        #sm_prices t with(nolock)
```

REFACTORING SQL

- Large stored procedures
- Break each table into domain models
- Similar to Active Record pattern
- Complicated joins could be done in the .Net layer



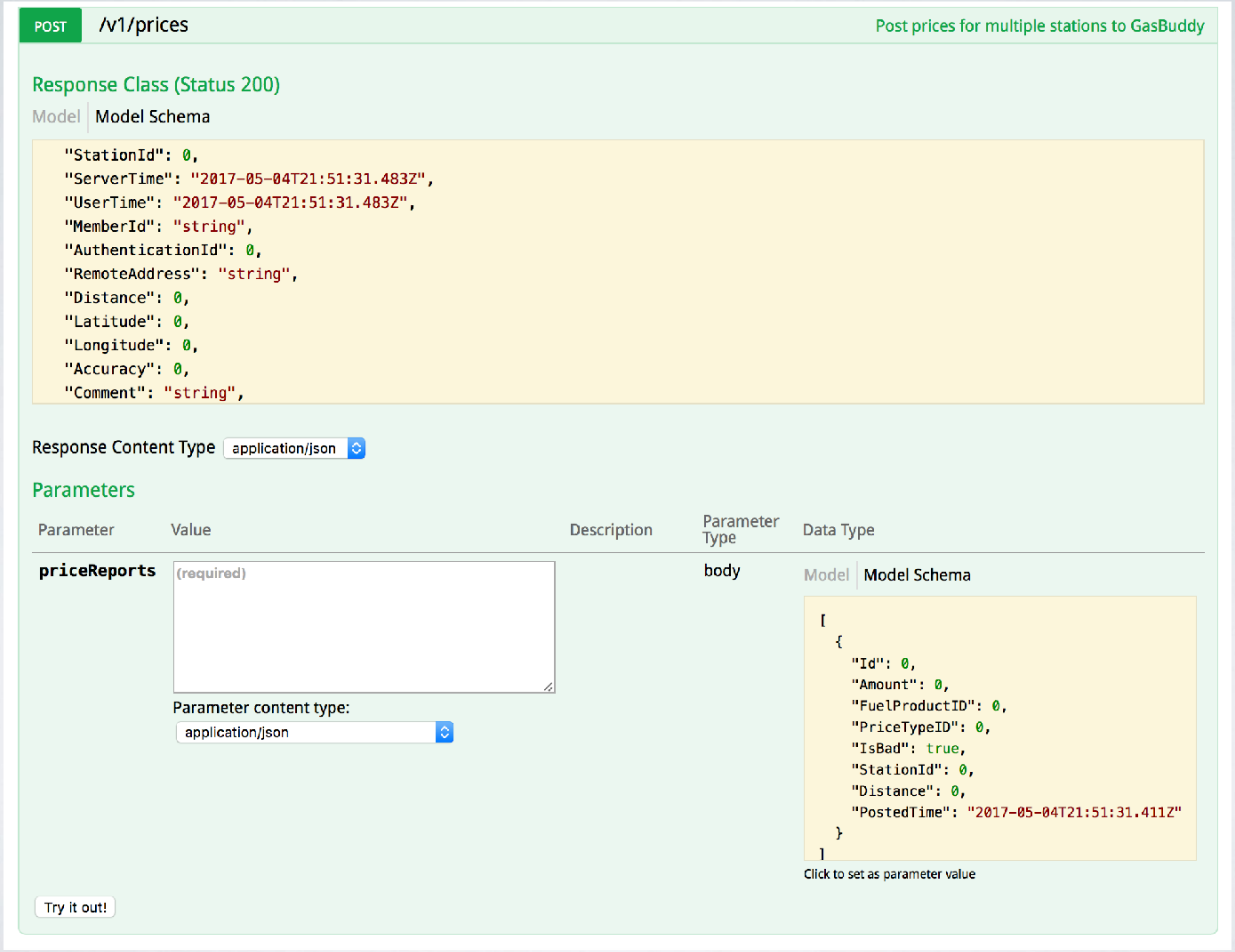
```
dbo.station_ratings
  Columns
    id (PK, int, not null)
    category_id (FK, int, not null)
    create_date (datetime, not null)
    review_id (FK, int, not null)
    sm_id (int, not null)
    value (tinyint, not null)
    score (tinyint, null)
  Keys
  Constraints
  Triggers
  Indexes
  Statistics
```



```
namespace GBLibrary.Models.Stations.Reviews
{

    /// <summary>
    /// A rating for a particular Station that is part of a Review.
    /// </summary>
    [DisplayName("station_ratings")]
    29 references | reganmeloche, 150 days ago | 2 authors, 2 changes | 1 work item
    public class Rating
    {

        5 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public int Id { get; set; }

        6 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public int CategoryId { get; set; }

        [DisplayName("stationmasterid")]
        5 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public int SmId { get; set; }

        8 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public int ReviewId { get; set; }

        4 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public int Value { get; set; }

        4 references | Christopher Johnson, 192 days ago | 1 author, 1 change
        public DateTime CreateDate { get; set; }

        4 references | reganmeloche, 150 days ago | 1 author, 1 change | 1 work item
        public int Score { get; set; }

    }
}
```

OPENAPI / SWAGGER

- Standard for making and documenting APIs
- For .Net we use Swashbuckle to make Swagger Client out of code
- We have used API clients in .Net, Node and Python



POST /v1/prices — Post prices for multiple stations to GasBuddy

Response Class (Status 200)

Model | Model Schema

```
  "StationId": 0,
  "ServerTime": "2017-05-04T21:51:31.483Z",
  "UserTime": "2017-05-04T21:51:31.483Z",
  "MemberId": "string",
  "AuthenticationId": 0,
  "RemoteAddress": "string",
  "Distance": 0,
  "Latitude": 0,
  "Longitude": 0,
  "Accuracy": 0,
  "Comment": "string",
```

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
priceReports	(required) Parameter content type: application/json		body	Model | Model Schema (see below) Click to set as parameter value

```
[
  {
    "Id": 0,
    "Amount": 0,
    "FuelProductID": 0,
    "PriceTypeID": 0,
    "IsBad": true,
    "StationId": 0,
    "Distance": 0,
    "PostedTime": "2017-05-04T21:51:31.411Z"
  }
]
```

REFACTORING RESULTS

- A price report can be made with an easy API call
- Price reporting takes less load on the database
- Added caching and logging (Redis and Logstash)

REBUILDING

"They did it by making the **single worst strategic mistake** that any software company can make: They decided to rewrite the code from scratch."

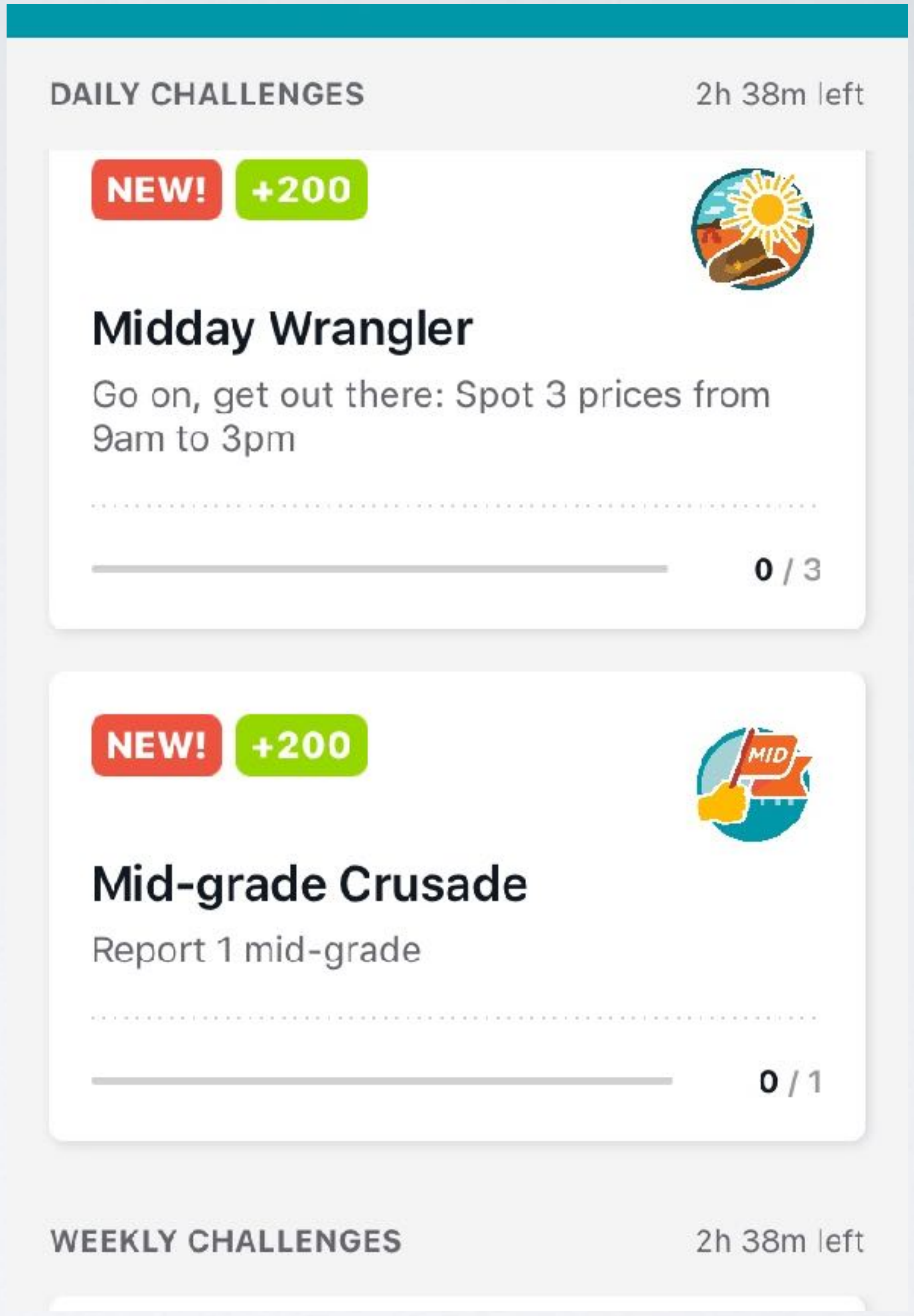
–Joel Spolsky
https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/

WHY REBUILD?

- System causes performance issues
- Doesn’t meet the needs of the business
- Code difficult / impossible to test or maintain

CHALLENGES

Reward for specific actions



DAILY CHALLENGES 2h 38m left

NEW! +200

Midday Wrangler

Go on, get out there: Spot 3 prices from 9am to 3pm

0 / 3

NEW! +200

Mid-grade Crusade

Report 1 mid-grade

0 / 1

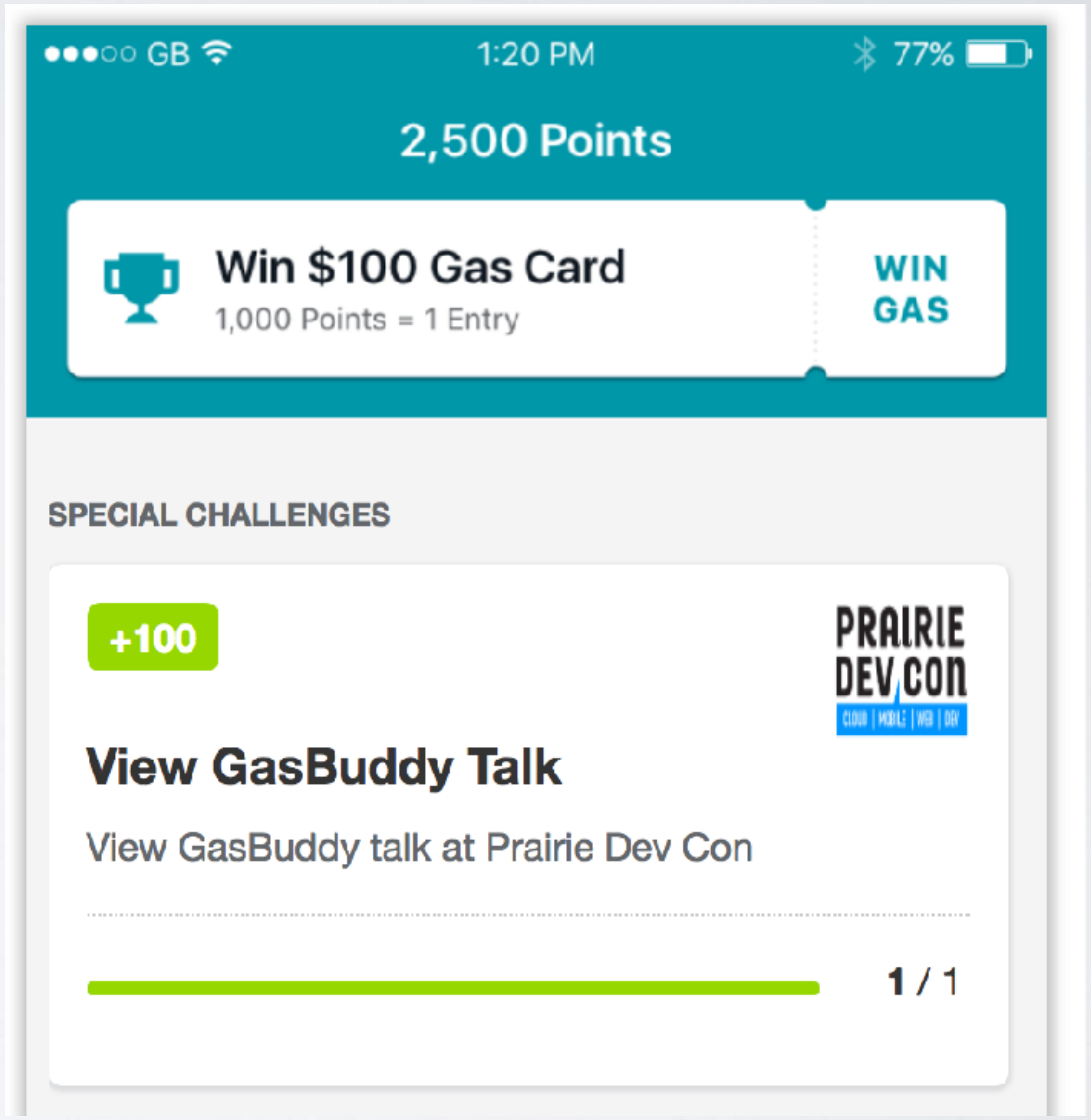
WEEKLY CHALLENGES 2h 38m left

OLD CHALLENGES

- System accounted for 25% DB CPU
- Written in a stored procedure
- Achievements weren’t flexible

CHALLENGES

- PostGres SQL on AWS
- Elastic Search for Geo
- .Net Middleware



2,500 Points

Win $100 Gas Card

1,000 Points = 1 Entry

WIN GAS

SPECIAL CHALLENGES

+100

View GasBuddy Talk

View GasBuddy talk at Prairie Dev Con

1 / 1

REMOVE

- Systems that are no longer necessary could be removed
- Old systems that have limited business use
- Analytics that don’t need to be on transactional system

REMOVE CASE STUDY

- Analytics we moved from our SQL server to Redshift
- Used DOMO as a front end
- Analytics and marketing / product reports not done on main database

RESULTS

- SQL server runs under 25% CPU
- We have an API
- Partially in the cloud

RECAP

- Refactor - Focus on the API
- Rebuild - Build system in the cloud or hybrid
- Remove - Legacy reporting system