Obsługa danych przestrzennych dwuwymiarowych
Sprawozdanie

Leon Ożóg

Czerwiec 2024

Contents

1 Założenia projektu

Celem projektu było stworzenie serwisu bazodanowego umożliwiającego użytkownikom przechowywanie danych przestrzennych dwuwymiarowych i dokonywanie na nich operacji w wydajny sposób. Interfejs miał udostępniać, miedzy innymi, możliwość wyznaczania odległości pomiędzy punktami, wyznaczenie pola określonego obszaru oraz określania czy punkt należy do danego obszaru. W celu implementacji powyższych funkcjonalności należało wykorzystać Common Language

Runtime User-Defined data Types. Stworzone Api zostało wykorzystane do zaprogramowania małej demonstracyjnej aplikacji webowej inspirowanej serwisem Google maps.

2 Wstęp teoretyczny

Wydajne operacje na dwuwymiarowych danych przestrzennych nie należą z pozoru do najłatwiejszych w implementacji, z powodu nieoczywistego sposobu w jaki powinniśmy indeksować dane, w celu wykorzystania istniejących technik optymalizacji zapytań bazodanowych. Jest to kwestia o tyle istotna, że ilość danych przetwarzana w dzisiejszych systemach wykorzystujących tego typu technologie, byłaby niemożliwa do przetworzenia w akceptowalnym czasie, jeżeli zmuszeni bylibyśmy stosować rozwiązania naiwne. Z pomocą przychodzą nam w tej kwestii metody implementujące algorytmy wypełniania wielowymiarowych przestrzeni krzywymi, które upraszczają je do mniejszej ilości wymiarów. To właśnie ta technika pozwala nam na wydajne indeksowanie wielowymiarowych danych, wyszukiwanie ich i przetwarzanie.

2.1 Geohash

Jednym z najpopularniejszych systemów tego typu jest stworzony w 2008 roku przez Gustava Niemeyer'a dostępny w domenie publicznej algorytm Geohash. Działa on na zasadzie enkodowania dwuwymiarowych koordynatów w postaci 12-elemntowego ciągu znakowego. Nie ma teoretycznego limitu co do długości ciągu znakowego, ale w praktyce wykorzystuje się jedynie te 12 elementowe. Po pierwsze z dlatego, że już taka długość pozawala na zapisywanie lokalizacji na Ziemi z dokładnością co do $\pm 1.2cm$, ale również z uwagi na to, że taka długość idealnie mieści się w 64 bitach pamięci.

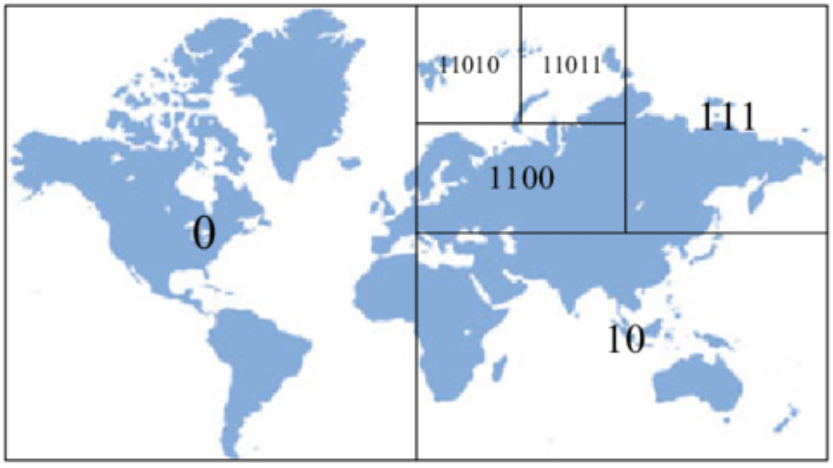

Figure 1: Sposób enkodowania pozycji przy pomocy Goehash

Pozycja w przestrzeni jest hierarchicznie dzielona naprzemiennie dla pozycji X, oraz dla Y, a potem bitowo zamieniana na ciąg znaków przy pomocy 32-elementowego systemu numerycznego Base32 (0123456789bcdefghjkmnpqrstuvwxyz).

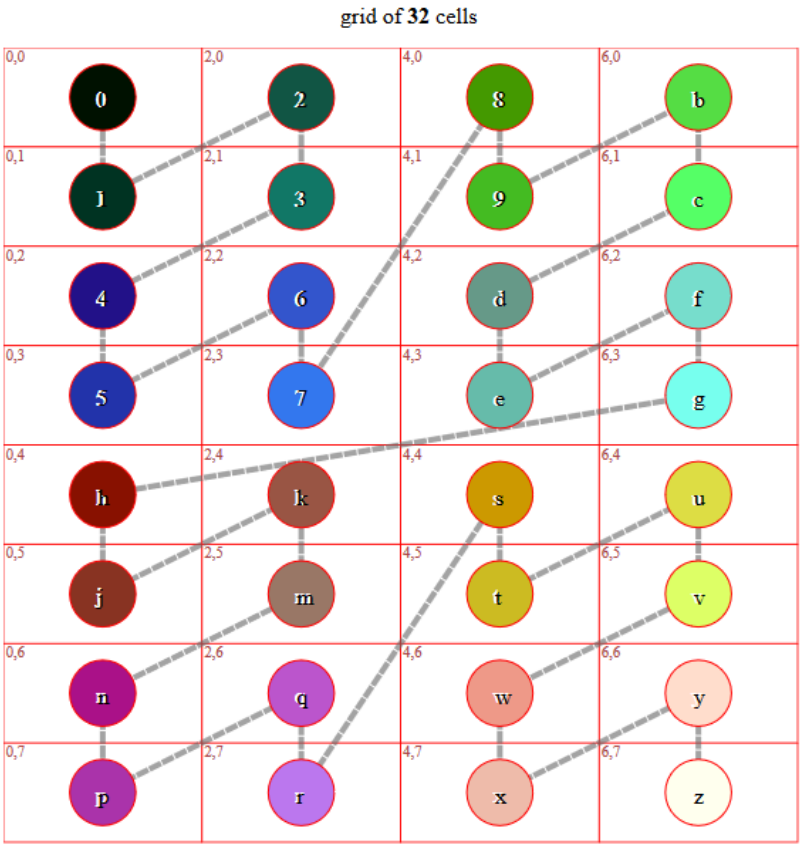

Figure 2: Zamiana dwuwymiarowej przestrzeni na przestrzeń jednowymiarową

Ten sposób reprezentacji, pozwala na indeksowanie pozycji i obszarów w bazach danych w wyjątkowo wydajny sposób. Geohash zapewnia, że każdy podprzestrzeń, może być przechowywany w pamięci w postaci skończonej ilości ciągłych obszarów, których ilość zależy tylko od dokładności kodowania jaką przyjmiemy, oraz tzw. "fault lines" czyli nagłych zmian bitu na wysokiej pozycji.
Kolejną zaletą wynikającą ze struktury zapisu Geohashy, jest to, że punkty położone obok siebie blisko w przestrzeni najczęściej znajdować się bedą również blisko siebie w pamięci - będą różnić się ostatnimi znakami w stringu.

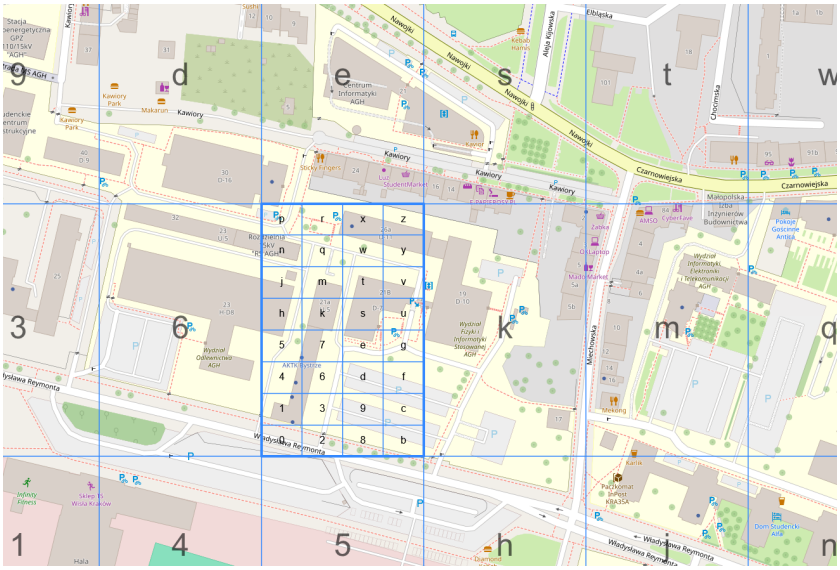

Figure 3: Budynek D7 WFIIS AGH można znaleźć pod geohashem u2yhv47

Data: Latitude and Longitude
Result: Geohash
Zainicjalizuj tablice zerami;
Zamień wysokość i szerokość geograficzną na przeplatany ciąg binarny:
while *długość ciągu binarnego* $< 5 \cdot$ *pożądana precyzja* **do**
  Podziel hierarchicznie szerokość lub długość geograficzną;
  Jeżeli punkt znajduje się po lewej stronie podziału, dopisz 0 do ciągu bitowego, a jeżeli po prawej, dopisz 1;
end
Zamień ciąg binarny w tablicy na ciąg znaków przy pomocy kodowania Base32;

Algorithm 1: Geohash Encoding

2.2 Geofencing - Even-odd rule

Even-odd rule jest techniką pozwalającą na określenie, czy punkt znajduje się wewnątrz obszaru, na podstawie ilości intersekcji jaką miałby z jego krawędziami promień wychodzący z tego punktu i zmierzający do nieskończoności. Jeżeli ilość ta jest nieparzysta, oznacza to, że punkt leży wewnątrz obszaru, a jeżeli parzysta, punkt leży na zewnątrz.

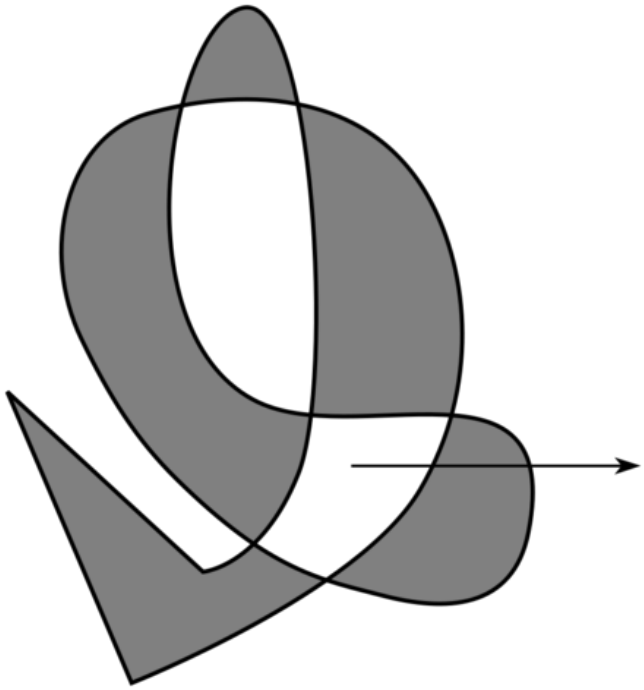

Figure 4: Even-odd rule

Algorytm ten sprawdza się wyśmienicie, natomiast jest niesamowicie obliczeniochłonny, w przypadku użycia go w zbiorze danych posiadającym wiele obszarów, o dużej ilości krawędzi. Punkt należy sprawdzić z każdym z nich i z każdą z ich krawędzi. Nawet jeżeli posłużymy się pewną optymalizacją, sprawdzając najpierw zawieranie w minimalnej ramce ograniczającej obszaru, algorytm ten nadal nie stanie się wydajnym w praktycznych zastosowaniach. Z pomocą po raz drugi przychodzi

nam system kodowania Geohash, który umożliwia nam indeksowanie danych w ten sposób, aby najpierw ograniczyć się do obszaru bazy danych, w której podejrzewamy znalezienie intersekcji, a dopiero potem używania bardziej dokładnych, acz czasochłonnych metod.

Czasami nawet warto ograniczyć się do wykrywania zawierania jedynie w dużych obszarach (np. wtedy kiedy użytkownik patrzy na mapę z dużym oddaleniem. wtedy nie warto wykrywać zawierania w poszczególnych budynkach). Pomocnym w tym okazuje się obliczanie obszaru zakreślonego przez poligon.

2.3 Obliczanie powierzchni poligonu - Shoelace formula

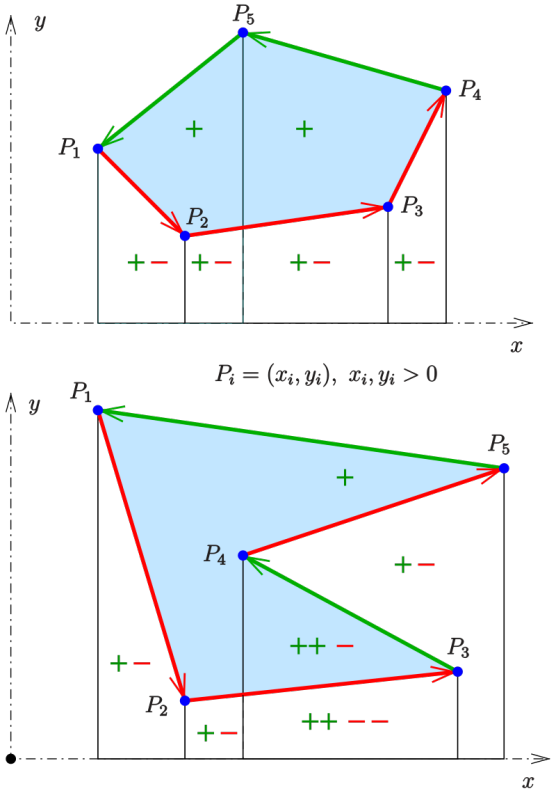

Figure 5: Shoelace formula

Formuła trapezów, znana również jako Shoelace formula, to matematyczny sposób obliczenia pola dowolnego wielokąta, znając jedynie współrzędne jego wierzchołków.

Działa ona na zasadzie sumowania pól, tworzonych przez trapezy wyznaczane przez skierowane krawędzie poligonu, tak że te kierujące się w pozytywnym kierunku osi, względem której tworzymy trapezy są odejmowane, a te które skierowane są w przeciwnym kierunku są dodawane. Ostatecznie skutkując wartością pola obszaru, z której następnie należy obliczyć wartość bezwzględną.

$$\mathrm{A} = \frac{1}{2}\left|\sum_{i=1}^{n}(x_i y_{i+1} - y_i x_{i+1})\right|$$

3 Zaimplementowane funkcjonalności

3.1 Geohash

Geohash jest typem wykorzystywanym przez wszystkie pozostałe typy w Api do tworzenia wydajnych indeksów w bazie danych.

- Konstruktory pozwalają na tworzenie obiektu na podstawie stringu jak i współrzędnych.
  `public Geohash(double x, double y, int precision = 12)`
  `public static Geohash Parse(SqlString s)`
  `public static Geohash Encode(double X, double Y, int precision)`

- Metoda ToString pozwala na uzyskanie zakodowanej wartości geohasha.
  `public override string ToString())`

- Metoda Decode pozwala na uzyskanie współrzędnych minimalnej ramki ograniczającej geohasha.
  `public double[] Decode()`

- Metoda CompareTo pozwala na wydajne porównywanie geohashy, kluczowe do efektywnego indeksowania.
  `public int CompareTo(object obj)`

- Metoda Includes sprawdza, czy jeden geohash znajduje się w drugim.
  `public bool Includes(Geohash other)`

- Metoda VisibleFrom sprawdza, czy jeden geohash może być widoczny z drugiego.
  `public bool VisibleFrom(Geohash other)`

- Metoda Common zwraca geohash o wspólnym prefiksie.
  `public Geohash Common(Geohash other)`

- Metoda size_x i size_y zwracają rzeczywistą wielkość obszaru zawartego w geohashu na mapie.
  `public static double size_x (int precision)`
  `public static double size_y (int precision)`

3.2 Point2d

Klasa Point2d reprezentuje podstawowy punkt na przestrzeni dwuwymiarowej. Jej interfejs automatycznie indeksuje jego położenie przy pomocy geohasha (ale dopiero wtedy kiedy jest on potrzebny, żeby uniknąć niepotrzebnych obliczeń). Aby w pełni wykorzystać możliwości optymalizacyjne, należy ustawić w tabeli index na polach typu Point2d.

- Konstruktory umożliwiają stworzenie punktu na podstawie string jak i współrzędnych.
  Konstruktor przyjmujący string, tworzy punkt na podstawie formatu *x,y*.
  `public Point2d(double x, double y)`
  `public static Point2d Parse(SqlString S)`

- Metoda ToString reprezentacji punktu w formie *x,y*
  `public override string ToString()`

- Metoda porównania pozwala na indeksowanie punktów w bazie danych.
  `public int CompareTo(object obj)`

- Metoda VisibleFrom sprawdza czy punkt widoczny jest z wyznaczonego prostokąta.
  `public bool VisibleFrom(double x1, double y1, double x2, double y2)`

- Metoda Dist oblicza odległość punktu od innego wyznaczonego punktu.
  `public double Dist(Point2d other)`

- Metoda SetXY pozwala na zmianę koordynatów punktu (inwalidując przy tym obliczony wcześniej geohash)
  `public void SetXY(double x, double y)`

3.2.1 UDF points_in_window

Tabelaryczna funkcja points_in_window() pozwala na efektywne wyszukiwanie punktów znajdujących się w prostokącie wyznaczonym przez dwa punkty. Wykorzystuje stworzony przez Point2d indeks.

3.3 Polygon2d

Polygon2d reprezentuje wielokąt na przestrzenie dwuwymiarowej i zawiera w sobie listę składającą się z objektów Point2d. Oprócz tego posiada w sobie pola odpowiedzialne za obliczanie jego obszaru oraz dwa punkty krańcowe, pozwalające na wydajne jego indeksowanie przy użyciu geohashy. Aby w pełni wykorzystać możliwości optymalizacyjne, należy ustawić w tabeli index na polach Polygon.GeohashMin, Polygon.GeohashMax oraz Polygon.GeohashArea. Przykład tego typu implementacji znajduje się w pliku *create_user.sql*.

- Konstruktory umożliwiają stworzenie poligonu na podstawie stringu jak i listy punktów. Konstruktor przyjmujący string, tworzy poligon na podstawie formatu $x_1, y_1; x_2, y_2; x_3, y_3; x_4, y_4; ....$
  `public Polygon2d(List<Point2d> points)`
  `public static Polygon2d Parse(SqlString s)`

- Poniższe pola umożliwiają indeksownie poligonu w bazie.
  `public Point2d BorderMin`
  `public Point2d BorderMax`
  `public Geohash GeohashMin`
  `public Geohash GeohashMax`

- Pole center udostępnia punkt znajdujący się po centrum poligonu.
  `public Point2d Center`

- W polu Area dostęna jest wartość obszaru ograniczanego przez poligon.
  `public double Area`

- Metody Includes pozwalają na sprawdzenie czy punkt leży wewnątrz poligonu.
  `public bool Includes(Point2d point)`
  `public bool IncludesNoGeohashCheck(Point2d point)`

  Poniższe metody pozwalają na dostęp do puntów składających się na wielokąt.
  `public void AddPoint(Point2d point)`
  `public void RemovePoint(Point2d point)`
  `public Point2d this[int index]`

3.3.1 UDF polygon_includes

Tabelaryczna funkcja polygon_includes() pozwala na efektywne wyszukiwanie poligonów, o obszarze większym niż zadany, w których znajduje się punkt. Wykorzystuje stworzone przez Polygon2d indeksy.

3.3.2 UDF polygons_in_window

Tabelaryczna funkcja polygons_in_window() pozwala na efektywne wyszukiwanie poiligonów, o obszarze większym niż zadany, widocznych w prostokącie wyznaczonym przez dwa punkty. Wykorzystuje stworzone przez Polygon2d indeksy.

4 Testy jednostkowe

Wszystkie przedstawione typy danych zostały dogłębnie przetestowane. Przy pomocy biblioteki C# nunit.

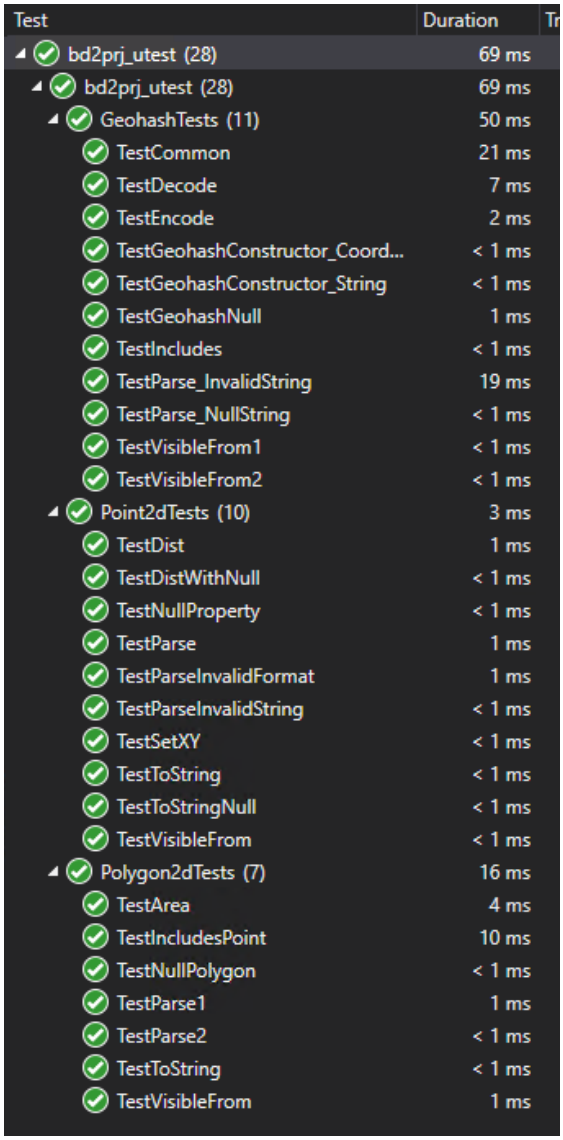

Figure 6: Przeprowadzone testy

5 Przykładowa aplikacja

Przykładowa aplikacja umożliwia tworzenie poligonów i punktów na płaszczyźnie przypominającej projekcje kuli ziemskiej. Przytrzymując przycisk myszy możemy poruszać kamerą, a kręcąc kółkiem przybliżać. Najeżdżając wskaźnikiem na poligon, wyświetlają się o nim dane przechowywane w bazie danych w postaci XML. Mamy też możliwość wyznaczenia odległości między punktami, po podaniu obu ich ID. Aby zaprezentować możliwości optymalizacyjne poligony przestają być pobierane jeżeli nie zajmują więcej niż 0.002 powierzchni ekranu. Tak samo wykrywane jest kiedy nie znajdują się one na ekranie. Można to zauważyć poruszając szybko myszką. Aplikacja została zaprogramowana przy użyciu serwera Node.js oraz JavaScript.

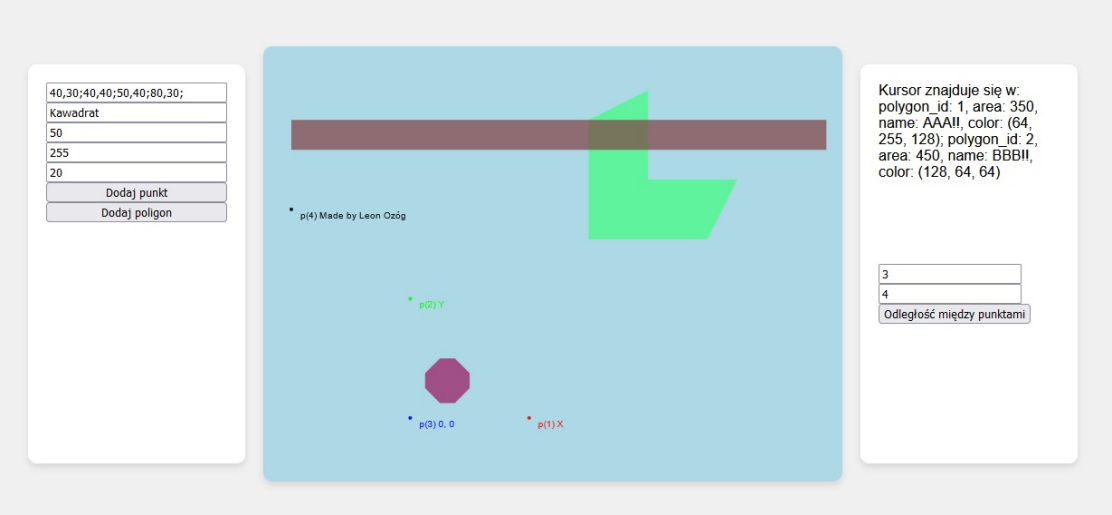

Figure 7: Przykładowa aplikacja

6 Kod źródłowy, jego uruchomienie i dokumentacja

Kod projektu znajduje się w archiwum przesłanym wraz ze sprawozdaniem. W katalogu bd2prj znajduje się projekt C# Visual Studio zawierający w sobie kod CLR UDR oraz testy z nim zwiazane. Aby skonfigurować baze danych należy uruchomić plik query, *baza.sql* znajdujący się w głównym katalogu, lub ręcznie skompilować UDT i następnie uruchomić plik query *create_user.sql*. W obu przypadkach można następnie wgrać dane testowe plikiem query *insert_test.sql*. Aplikacja napisana w Node.js znajduje się w katalogu app. Abu uruchomić aplikację należy pobrać wszystkie wymagane pakiety komendą *npm install* oraz uruchomić ją komendą *npm start*. Strona internetowa powinna uruchomić się na porcie 3030. Dokumentacja znajduje się w plikach z kodem.

7 Podsumowanie i wnioski

Projekt się udał. Dane przestrzenne to niezwykle interesująca i ważna dziedzina typów danych bazodanowych.

8 Literatura

https://en.wikipedia.org/wiki/Geohash
https://geohash.softeng.co/u2yhv47
https://en.wikipedia.org/wiki/Even%E2%80%93odd_rule
https://en.wikipedia.org/wiki/Shoelace_formula
https://learn.microsoft.com/en-us/sql/relational-databases/clr-integration-database-objects-user-defined-types/clr-user-defined-types?view=sql-server-ver16
https://learn.microsoft.com/en-us/sql/relational-databases/clr-integration-database-objects-user-defined-types/working-with-user-defined-types-in-sql-server?view=sql-server-ver16