

Spis treści

1	Wstęp	4
1.1	Uzasadnienie wyboru tematu	4
1.2	Problematyka i zakres pracy	6
1.3	Cele pracy	7
1.4	Metoda badawcza	8
1.4.1	Studia literaturowe	8
1.4.2	Analiza istniejących rozwiązań	9
1.4.3	Stworzenie własnej aplikacji szkieletowej	9
1.4.4	Analiza porównawcza oraz testy	9
1.5	Przegląd literatury w dziedzinie	9
1.5.1	Literatura dotycząca języka C++ oraz Qt	9
1.5.2	Literatura dotycząca języka SQL	10
1.5.3	Literatura dotycząca mapowania obiektowo-relacyjnego	10
1.6	Układ pracy	10
2	Zagadnienia teoretyczne	12
2.1	Architektura warstwowa	12
2.2	Trwałość danych	13
2.3	Relacyjne bazy danych	13
2.4	Programowanie obiektowe	14
2.5	Wykorzystanie SQL w C++	15
2.6	Niedopasowanie paradygmatów	15
2.7	Koszt niedopasowania	16
2.8	Mapowanie obiektowo-relacyjne	17
2.9	Aplikacje szkieletowe	19
3	Analiza istniejących rozwiązań	21
3.1	Kryteria analizy	21
3.2	Porównanie istniejących rozwiązań	22

3.2.1	Biblioteka QxOrm	22
3.2.2	Biblioteka Debea	25
4	Aplikacja szkieletowa Qubic	27
4.1	Moduły tworzonej aplikacji	27
4.2	Analiza wymagań	28
4.2.1	Wymagania funkcjonalne	28
4.2.2	Wymagania нефункционалне	28
4.3	Projekt	29
4.3.1	Rodzaj aplikacji	29
4.3.2	Diagram klas	29
4.3.3	Wzorce projektowe	31
4.3.4	Środowisko programistyczne	31
4.3.5	System kontroli wersji	31
4.4	Implementacja	32
4.4.1	Połączenie z bazą danych	33
4.4.2	Interfejs CRUD	33
4.4.3	Interfejs do tworzenia zapytań	40
4.4.4	Metody dostępu do powiązanych danych	40
4.5	Podręcznik użytkownika	43
4.6	Przykładowa aplikacja wykorzystująca Qubica	44
4.7	Analiza Qubica	50
4.8	Perspektywy rozwoju Qubica	51
5	Podsumowanie	53
5.1	Dyskusja wyników	53
5.2	Perspektywy rozwoju pracy	53
	Bibliografia	54
	Spis rysunków	56
	Spis tabel	57
	Spis listingów	58

Rozdział 1

Wstęp

1.1 Uzasadnienie wyboru tematu

Wraz z upływem czasu postęp technologiczny ma wpływ na życia co raz szerszej rzeszy ludzi na całym świecie. Niezliczone ilości urządzeń zagościły na stałe w domach i mało kto wyobraża sobie bez nich swoje życie. Zaczynając od artykułów gospodarstwa domowego, a kończąc na elektronice użytkowej do której zaliczają się komputery, telewizory czy też smartfony¹. Wszystkie te urządzenia mają na celu ułatwianie życia swoim użytkownikom.

W parze z licznymi zaletami urządzeń elektronicznych idą jednak pewne wady. Jedną z istotniejszych jest wpływ czasu spędzanego przed różnego rodzaju wyświetlaczami na zdrowie. Badania przeprowadzone na bazie danych Nielsen Audience Measurement pokazują, że przeciętny Polak spędza dziennie średnio 4,5 godziny przed ekranem telewizora². Nie oznacza to jednak, że przez cały ten czas ogląda on telewizję. Oglądanie filmów z dysku komputera, za pomocą serwisów VOD³ czy granie na konsoli także są wliczone w ten czas. Gdyby jednak dodać do tego czas spędzony przed ekranem smartfona czy też komputera wynik byłby zapewne dwukrotnie większy.

Pogorszenie wzroku czy też wysychanie gałki ocznej są wymieniane jako najczęstsze skutki zbyt dużej ilości czasu spędzanego przed ekranem. Poza próbą jego ograniczenia, jedną z częstszych porad jest próba zmniejszenia kontrastu pomiędzy ekranem a jego otoczeniem.

¹ Przenośne urządzenia łączące w sobie zalety telefonów komórkowych oraz przenośnych komputerów (z ang. smartphone).

² Badania zostały przeprowadzone z uwzględnieniem osób powyżej 4 roku życia w okresie od stycznia do czerwca 2015 roku [?].

³ Wideo na życzenie (z ang. video on demand).

Do celów niniejszej pracy należy złożenie i oprogramowanie systemu oświetlenia, który ma rozszerzać obraz widziany na ekranie na jego otoczenie. Poza zmniejszeniem kontrastu, a więc aspektem zdrowotnym, system ma także na celu zwiększyć wrażenia wizualne dostarczane przez oglądany obraz.

System oświetlenia składa się z taśmy diod elektroluminescencyjnych⁴ podłączonych do mikrokontrolera Arduino Uno, który z kolei ma współpracować z komputerem z zainstalowanym systemem operacyjnym MacOS. Oprogramowanie mikrokontrolera, którego zadaniem jest sterowanie diodami zostało przygotowane w języku Arduino, natomiast aplikacja kontrolująca cały system przeznaczona na komputer z systemem MacOS została napisana w języku Swift. Wybór języków jest ściśle związany z koniecznością uzyskania jak najlepszej wydajności oraz użyciem najnowszych technologii.

Podobne systemy oświetlenia dostępne są już od pewnego czasu na rynku, jednak to właśnie nowoczesne technologie, prostota wykonania i niskie koszty powinny uczynić z Lightning, bo taką nazwę otrzymał projekt, pełnowartościowego konkurenta.

1.2 Problematyka i zakres pracy

1.3 Cele pracy

1.4 Metoda badawcza

1.5 Przegląd literatury w dziedzinie

1.6 Układ pracy

⁴ LED (z ang. light-emitting diode).

Rozdział 2

Zagadnienia teoretyczne

Rozdział 3

Analiza istniejących rozwiązań

3.1 Kryteria analizy

3.2 Porównanie istniejących rozwiązań

Rozdział 4

System oświetlenia Lightning

4.1 Komponenty systemu

4.2 Moduły oprogramowania

4.2.1 Oprogramowanie przeznaczone na mikrokontroler

4.2.2 Aplikacja przeznaczona na komputer

4.3 Analiza wymagań

4.3.1 Wymagania funkcjonalne

W celu stworzenia jak najatrakcyjniejszego system oświetlenia podczas projektowania Lightning przyjęto poniższe wymagania funkcjonalne:

- System musi udostępniać tryb przechwytywania rozszerzający obraz wyświetlany na ekranie na jego otoczenie za pomocą diod elektroluminescencyjnych.
- System musi posiadać tryb animacji za pomocą diod elektroluminescencyjnych. Powinien być on łatwy do rozszerzenia o kolejne animacje.
- System musi być konfigurowalny z poziomu aplikacji sterującej. Konfiguracji podlegać musi co najmniej liczba i rozmieszczenie diod za ekranem, a także wykorzystywany port szeregowy.
- Projekt musi być odpowiednio udokumentowany. Dokumentacja powinna ułatwiać szybkie skonfigurowanie oraz uruchomienie systemu.

4.3.2 Wymagania niefunkcjonalne

Do wymagań niefunkcjonalnych postawionych przed projektowanym systemem należą:

- System musi działać płynnie, a więc liczba osiągniętych klatek na sekundę powinna być jak największa nawet przy dużych rozdzielczościach przechwytywanego ekranu.
- Korzystanie z aplikacji sterującej powinno być jak najbardziej intuicyjne, a użytkownik powinien mieć do wskazówek dotyczących jej interfejsu.
- Oprogramowanie mikrokontrolera powinno oferować jak najprostszy interfejs i zawierać jak najmniej logiki sterującej.
- Pamięć na obu urządzeniach sterujących powinna być odpowiednio zarządzana, niedopuszczalne są wycieki pamięci czy też zapętlenia programu.

4.4 Projekt



Rysunek 4.1: Logo projektu

4.5 Implementacja

4.6 Podręcznik użytkownika

4.7 Przykładowa implementacja animacji

4.8 Analiza projektu

4.9 Perspektywy rozwoju projektu

Rozdział 5

Podsumowanie

5.1 Dyskusja wyników

5.2 Perspektywy rozwoju pracy

Bibliografia

- [1] <http://www.wirtualnemedial.pl/arttykul/coraz-dluzej-ogladamy-telewizje-najwiecej-czasu-przed-szklanym-ekranem-spedzaja-seniorzy- raport>. Data dostępu – 15.01.2017.

Spis rysunków

1.1	Wykres przedstawiający popularność mapowania obiektowo-relacyjnego na przestrzeni czasu w wybranych językach programowania	5
1.2	Schemat współpracy modułów tworzonej aplikacji szkieletowej . .	5
2.1	Warstwowa architektura aplikacji	12
2.2	Mapowanie obiektowo-relacyjne	17
3.1	Biblioteka QxOrm	22
3.2	Biblioteka Debea	25
4.1	Logo tworzonej aplikacji szkieletowej	29
4.2	Diagram klas	30
4.3	Przejrzysty widok systemu GitHub Issues	32
4.4	Przejrzysty widok systemu GitHub Milestones	32
4.5	Schemat blokowy metody zapisującej obiekty w bazie danych . . .	36
4.6	Schemat blokowy metody aktualizującej obiekty w bazie danych . .	37
4.7	Schemat blokowy metody ładującej obiekty z bazy danych	38
4.8	Schemat blokowy metody usuwającej obiekty z bazy danych	39

Spis tablic

3.1	Analiza biblioteki QxOrm	23
3.2	Analiza biblioteki Debea	26
4.1	Analiza aplikacji szkieletowej Qubic	50

Spis listingów

3.1	Przykładowy plik nagłówkowy klasy korzystającej z QxOrm	23
3.2	Przykładowy plik klasy korzystającej z QxOrm	24
3.3	Przykładowy plik klasy korzystającej z QxOrm	24
3.4	Przykładowy użycie Debei	26
4.1	Ciało metody dostępu do powiązanych danych (jeden do wielu) . . .	41
4.2	Zapytanie dostępu do powiązanych danych (jeden do wielu)	41
4.3	Ciało metody dostępu do powiązanych danych (wiele do wielu) . . .	41
4.4	Zapytanie dostępu do powiązanych danych (wiele do wielu)	42
4.5	Tworzenie zapytań Qubica	44
4.6	Plik tworzący przykładową bazę danych	44
4.7	Plik przedstawiający przykładową aplikację korzystającą z Qubica .	46
4.8	Plik wynikowy przykładowej aplikacji korzystającej z Qubica . . .	49