AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa magisterska

Budowa aplikacji www wspierającej tryb off-line przy użyciu HTML5
Building offline enabled web application with HTML5

Autor: *Aleksander Nowak, Konrad Oboza*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr inż. Paweł Skrzyński*

Kraków, 2014

Oświadczamy, świadomi odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonaliśmy osobiście i samodzielnie i nie korzystaliśmy ze źródeł innych niż wymienione w pracy.

Składamy serdeczne podziękowania Panu dr Pawłowi Skrzyńskiemu za cenne wskazówki i pomoc w trakcie pisania pracy.

Spis treści

1. Wstęp

Pozycja sieci Internet jako podstawowego i powszechnego medium komunikacyjnego jest niezachwiana od momentu jej powstania. Obecnie niemal wszystkie aplikacje dokonują synchronizacji za pomocą protokołów internetowych bez względu na platformę, na którą są przeznaczone. Dynamicznie rozwijające się standardy związane z protokołem HTTP (Hypertext Transfer Protocol) czyniące przeglądanie zasobów globalnej sieci łatwym i intuicyjnym za pośrednictwem każdego rodzaju urządzenia, potężne rozwiązania chmurowe oferujące platformy (PaaS), infrastruktury (IaaS) lub oprogramowanie (SaaS) jako usługi, czy szybka i przenośna bankowość elektroniczna to wybrane powody niesłabnącego zainteresowania rozwiązaniami internetowymi.

Tendencje rozwoju współczesnego oprogramowania warunkują konieczność znalezienia rozwiązania w sytuacjach braku dostępu do sieci Internet. Obejmuje ono metody przechowywania danych lokalnie z użyciem przeglądarki internetowej, sposoby ich replikacji i synchronizacji oraz detekcję połączenia.

Niniejsza praca dokonuje przeglądu wiodących metod lokalnego gromadzenia i przetwarzania danych użytkownika oraz obejmuje stworzenie aplikacji umożliwiającej pracę w trybach online oraz offline wraz z pełną synchronizacją danych pomiędzy klientem a serwerem wraz z bazą danych.

Postaramy się ustalić które z nowoczesnych technologii najlepiej radzą sobie z zachowaniem integralności w razie braku dostępu do sieci Internet.

Struktura niniejszej pracy jest następująca. Rozdział 1 przedstawia koncepcję i cele projektu. W rozdziale 2 dokonano analizy wymagań składowych projektu i przedstawiono wykorzystane technologie oraz paradygmaty. Rozdział 3 przedstawia założenia projektowe poszczególnych komponentów wraz z objaśnieniem wykorzystanych algorytmów. Rozdział 4 poświęcony jest przedstawieniu szczegółów implementacji. W rozdziale 5 zamieszczono krótką instrukcję instalacji i obsługi aplikacji. Rozdział 6 stanowi podsumowanie osiągniętych rezultatów.

1.1. Cel pracy

Celem niniejszej pracy jest przegląd metod lokalnego przechowywania danych z użyciem pamięci podręcznej przeglądarki oraz projekt i implementacja aplikacji wspierającej tryby offline oraz online w oparciu o technologię HTML5.

1.2. Dziedzina problemu

Aplikacja OffCalendar stanowi przykład praktycznego wykorzystania trybu offline i metod przechowywania lokalnego oraz synchronizacji. Wiodącą technologią zastosowaną w projekcie jest HTML5 oraz JavaScript po stronie aplikacji klienckiej oraz PHP i MySQL po stronie serwerowej.

Głównym celem projektu jest dostarczenie praktycznego i intuicyjnego rozwiązania będącego owocem gruntownej analizy dostępnych technologii lokalnego przechowywania danych użytkownika.

Dane gromadzone lokalnie przez użytkownika pozbawionego dostępu do sieci Internet będą automatycznie synchronizowane z wersją bazy danych na serwerze w momencie poprawnej detekcji i ustalenia połączenia ponownie. Całość systemu opiera się na założeniach architektury REST (*Representational State Transfer*) wywodzonej ze specyfikacji protokołu HTTP.

Działanie aplikacji klienckiej polega na organizowaniu danych w formie wydarzeń w obrębie przestrzeni roboczej przedstawionej w postaci dni miesiąca z podziałem godzinowym. Użytkownik będzie miał możliwość dodawania kolejnych pozycji wraz z tytułem, opisem, czasem rozpoczęcia oraz czasem zakończenia. Zbliżające się terminy wydarzeń będą sygnalizowane za pomocą systemu powiadomień (notyfikacji), który jest w pełni personalizowalny (częstość otrzymywania powiadomień, ich treść, wyprzedzenie czasowe etc.).

Zadaniem części serwerowej systemu będzie walidacja oraz przechowywanie/pobieranie danych z bazy danych MySQL.

Aplikacja OffCalendar z założenia stanowi intuicyjny i niezawodny system umożliwiający organizowanie wydarzeń na różnych urządzeniach oraz bez względu na stan połączenia internetowego. Zapewniają to technologie przechowywania danych lokalnie, które zostały wyłonione na drodze wnikliwej analizy oraz algorytmy synchronizacji, detekcji oraz replikacji opracowane na potrzeby niniejszej pracy.

2. Analiza techniczna i analiza wymagań

W rozdziale tym zostaną dokładnie przedstawione oraz porównane mechanizmy lokalnego przechowywania danych po stronie przeglądarki internetowej. Omówiony zostanie również problem dostępności aplikacji w trybie offline oraz używania aplikacji bez dostępnego połączenia internetowego i późniejszej synchronizacji danych.

Ponadto, w rozdziale zostaną przedstawione założenia systemu oraz wymagania które aplikacja powinna spełniać, a także narzędzia oraz technologie które pozwolą na wykonanie aplikacji zgodnie z założonymi wymaganiami.

2.1. Lokalne przechowywanie danych

Głównym wymogiem technicznym web aplikacji wspierającej tryb offline jest możlwość użytkowania jej bez konieczności połączenia z serwerem. Do realizacji tego wymogu niezbędne jest przechowywanie danych po stronie klienta - przeglądarki internetowej.

Przechowywanie danych po stronie klienta zawsze było mocną stroną aplikacji desktopowych. W przypadku web aplikacji, przez bardzo długi okres czasu ta funkcjonalność była ograniczona do tworzenia ciasteczek (cookie).

Cookie to niewielkie ilości danych każdorazowo przesyłane do serwera przy przeglądaniu danej witryny. Wsparcie przeglądarek dla plików cookie rozpoczęło się w 1995 roku i do drugiej dekady XXI wieku pliki cookie były jedyną szeroko wspieraną metodą przechowywania danych po stronie klienta. Pliki cookie posiadają kilka zasadniczych wad:

- Są przesyłane przy każdym zapytaniu HTTP kierowanym do serwera - niepotrzebnie zwiększenie transferu danych,
- Pliki cookie są przesyłane do serwera niezależnie od użycia protokołu SSL - ryzyko podsłuchania ruchu sieciowego i wycieku danych,
- Rozmiar pliku cookie nie może przekraczać 4KB - brak możliwości przechowywania większych ilości informacji.

W przypadku potrzeby uniknięcia ograniczeń cechujących ciasteczka, programista ma do wyboru kilka nowoczesnych metod przechowywania danych po stronie klienta, bazujących na specyfikacji

HTML5: WebSQL, Web Storage, IndexedDB, File API. Metody te pozwalają na przechowywanie znacznie większych ilości danych oraz są manualnie przesyłane do serwera, co pozwala na optymalizację zapytań oraz większe bezpieczeństwo danych.

Jednym z kluczowych aspektów w przypadku wyboru metody lokalnego przechowywania danych w web aplikacji jest lista przeglądarek które wspierają daną metodę. O ile w przypadku plików cookie możemy mówić o pełnym wsparciu, o tyle poszczególne metody HTML5 posiadają zróżnicowane poziomy wsparcia zarówno na przeglądarkach desktopowych jak i mobilnych.

2.2. Mechanizmy lokalnego przechowywania danych

Aby budowa aplikacji www wspierającej tryb offline była możliwa, konieczny jest dobór odpowiedniego mechanizmu lokalnego przechowywania danych. Poniżej, pod względem funkcjonalności, łatwości użytkowania oraz wsparcia na konkretnych przeglądarkach, przeanalizowane zostały obecnie dostępne metody lokalnego przechowywania danych. Z uwagi na niezgodność z wymaganiami funkcjonalnymi aplikacji, pliki cookie zostały pominięte w poniższej analizie.

2.2.1. HTML5 WebSQL Database

WebSQL Database jest mechanizmem, który do przechowywania, odczytu oraz aktualizacji danych po stronie klienta wykorzystuje strukturalny język zapytań SQL. Metoda działania mechanizmu bazuje na specyfikacji bazy SQLite. Mechanizm WebSQL posiada zarówno zalety przypisywane relacynym bazom danych oraz kilka zalet cechujących metody lokalnego zapisu danych:

- Transakcyjność,
- Łatwość w utrzymaniu integralności danych,
- Dobra wydajność w przypadku wyszukiwania zbioru elementów,
- Szerokie spektrum zastosowań,
- Rozmiar bazy danych ograniczony jedynie dostępną przestrzenią dyskową klienta,
- Asynchroniczne wykonywanie zapytań - interfejs użytkownika nie jest blokowany do czasu zwróceniu rezultatu przez bazę danych.

Mechanizm WebSQL sprawdzi się dobrze w przypadku z góry zdefiniowanych struktur danych, które nie muszą być często zmieniane. W przypadku konieczności zmiany struktury danych, wymagane jest każdorazowe napisanie skryptu wykonującego zapytania SQL które pozwolą na aktualizację struktury bazy danych.

WebSQL jest metodą posiadającą umiarkowane wsparcie wśród popularnych przeglądarek internetowych. Poniżej znajduje się tabela przedstawiające wsparcie przeglądarek dla mechanizmu WebSQL zarówno na systemach desktopowych jak i mobilnych:

Przeglądarka	Wsparcie (od wersji)
Google Chrome	4.0
Internet Explorer	brak
Mozilla Firefox	brak
Safari	3.1
Android Browser	2.1
Chrome for Android	35.0
iOS Safari	3.2

Tablica 2.1: Wsparcie mechanizmu WebSQL Database w przeglądarkach internetowych.

Przy wyborze mechanizmu WebSQL jako metody przechowywania danych po stronie klienta należy wziąć pod uwagę fakt, że specyfikacja standardu nie jest dalej rozwijana. Z tego względu, przy wyborze tej technologii należy liczyć się z możliwością utraty wsparcia na przeglądarkach, które w chwili obecnej je zapewniają. Dodatkowo, przeglądarki które w tym momencie nie oferują wsparcia dla tego mechanizmu, nie dodadzą go przy premierze kolejnych ich wersji.

Biorąc pod uwagę powyższe informacje, użytkowanie aplikacji www bazującej na mechaniźmie WebSQL po aktualizacji przeglądarki do nowszej wersji może stać się niemożliwe. Stoi to w sprzeczności z wymaganiami funkcjonalnymi aplikacji, dlatego mechanizm ten nie zostanie wykorzystany przy budowie aplikacji wspierającej tryb offline.

2.2.2. HTML5 Web Storage

HTML5 Web Storage, znany również jako DOM Storage, jest mechanizmem, który przechowuje dane w tablicy asocjacyjnej w której klucze oraz wartości są ciągami tekstowymi. W zależności od przeglądarki internetowej, maksymalny rozmiar danych możliwy do zapisu wynosi od 5 do 35 megabajtów danych.

Web Storage udostępnia dwa obszary składowania danych różniące się zasięgiem oraz cyklem życia:

– *localStorage* - dane są dostępne we wszystkich oknach oraz kartach danej przeglądarki, dane nie są usuwane po zamknięciu przeglądarki,

– *sessionStorage* - dane dostępne w obrębie jednej karty przeglądarki, usuwane po jej zamknięciu.

Największymi zaletami mechanizmu Web Storage jest kompleksowe wsparcie na wszystkich obecnie popularnych przeglądarkach internetowych, jak również łatwość jego obsługi - pobranie wartości danego elementu następuje synchronicznie poprzez odwołanie się do globalnego obiektu localStorage lub sessionStorage wraz z podaniem w notacji tablicowej klucza - ciągu znakowego - identyfikującego dany element.

Prostota mechanizmu Web Storage ma również swoje wady. Największą z nich jest brak możliwości przechowywania zaawansowanych struktur danych w formacie innym niż łańcuch znaków. W takim

Przeglądarka	**Wsparcie (od wersji)**
Google Chrome	4.0
Internet Explorer	8.0
Mozilla Firefox	3.5
Safari	4.0
Android Browser	2.1
Chrome for Android	35.0
iOS Safari	3.2

Tablica 2.2: Wsparcie mechanizmu Web Storage w przeglądarkach internetowych.

przypadku konieczna jest serializacja danych przy zapisie oraz ich deserializacja przy odczycie, co w przypadku dużych obiektów może doprowadzić do problemów z wydajnością.

Dużym ograniczeniem metody jest również brak możliwości efektywnego przeszukiwania danych oraz ich indeksowania. W większości przypadków prowadzi to do konieczności iteracji po wszystkich zdefiniowanych elementach. Synchroniczność zapytań oraz konieczność deserializacji otrzymanych danych skutecznie obniżają szybkość wyszukiwania, szczególnie w przypadku dużej ilości przechowywanych obiektów.

Zarówno konsorcjum W3C zajmujące się rozwojem standardu Web Storage, jak i firmy zajmujące się tworzeniem przeglądarek internetowych są zainteresowane umożliwieniem przechowywania bardziej zaawansowanych struktur danych. Pozwoliłoby to na efektywne zastosowanie Web Storage w przypadku złożonych obiektów. Biorąc pod uwagę cykl życia przeglądarek internetowych oraz średni czas potrzebny użytkownikowi na jej aktualizację, uzyskanie szerszego wsparcia dla tej aktualizacji może zająć kilka lat.

Mechanizm Web Storage posiada pełne wsparcie w obecnych klientach sieci www, jednak nie posiada on wydajnej metody przeszukiwania przechowywanych danych, przez co jego wykorzystanie przy budowie aplikacji www wspierającej tryb offline jest niezalecane na tym etapie rozwoju mechanizmu.

2.2.3. HTML5 IndexedDB

Indexed Database jest mechanizmem, który powstał na bazie Web Storage oraz WebSQL, łączy cechy obydwu specyfikacji. IndexedDB przechowuje obiekty języka JavaScript w magazynach obiektów (*objectStores*) które są odpowiednikami tabel w relacyjnej bazie danych.

Każdy rekord przetrzymywany w magazynie musi posiadać unikalny klucz główny który może zostać zdefiniowany przez użytkowika lub, w przeciwnym wypadku, zostanie utworzony automatycznie. W celu optymalizacji przeszukiwania zbioru danych, IndexedDB umożliwia zdefiniowanie indeksu na dowolnym z atrybutów przechowywanych obiektów. Znacząco skraca to czas wyszukiwania oraz ilość potrzebnych do tego zasobów.

Podobnie jak w przypadku mechanizmu WebSQL, IndexedDB wykonuje zapytania asynchronicznie,

Przeglądarka	Wsparcie (od wersji)
Google Chrome	23.0
Internet Explorer	10.0
Mozilla Firefox	10.0
Safari	8.0
Android Browser	4.4
Chrome for Android	35.0
iOS Safari	8.0

Tablica 2.3: Wsparcie mechanizmu IndexedDB w przeglądarkach internetowych.

pozwalając na interakcję użytkownika z aplikacją podczas wykonywania zapytań. Mechanizm wspiera również wersjonowanie bazy danych - w przypadku konieczności aktualizacji bazy do nowszej wersji wykonywane są zdefiniowane przez programistę instrukcje, które powinny zapewnić spójność poprzez aktualizację danych znajdujących się w poprzedniej wersji magazynów obiektów.

Ilość miejsca, jaka może zostać przeznaczona na zasoby przechowywane w IndexedDB jest nieograniczona. W przypadku przekroczenia granicznego rozmiaru bazy danych, indywidualnie ustalanego dla danej przeglądarki internetowej, użytkownik musi wyrazić zgodę na zniesienie ograniczenia ilości miejsca przeznaczonego na magazyny obiektów.

Indexed Database posiada dobre wsparcie wśród nowszych wersji popularnych przeglądarek internetowych.

IndexedDB łączy najlepsze funkcjonalności WebSQL oraz Web Storage. Biorąc pod uwagę dobre wsparcie wśród przeglądarkek, szybkie czasy wyszukiwania elementów przy użyciu indeksów oraz możliwość przechowywania obiektów bez konieczności ich serializacji, mechanizm ten spełnia wszystkie wymagania aplikacji www wspierającej tryb offline.

2.2.4. HTML5 Filesystem API

Filesystem API jest mechanizmem który umożliwia operacje na plikach znajdujących się w wydzielonej przestrzeni na dysku twardym użytkownika. Aby zapis na dysku był możliwy, użytkownik musi najpierw upoważnić do tego aplikację, co odbywa się poprzez zaakceptowanie okna dialogowego wyświetlającego żądanie przechowywania konkretnej ilości danych, jaką dana aplikacja zamierza wykorzystać.

Lista funkcji udostępnianych przez Filesystem API obejmuje zarówno operacje na plikach - zapis, odczyt pliku, odczyt metadanych pliku, kopia do wskazanej lokacji, oraz narzędzia do tworzenia drzewa katalogów oraz ich przeszukiwania. Pliki przechowywane na wydzielonej przestrzeni dyskowej mogą później zostać osadzone jako elementy strony www - jedna z metod Filesystem API zwraca pełny adres URL do danego zasobu.

Pomimo wysokiej użyteczności mechanizmu Filesystem API, wsparcie przeglądarkek ograniczna

Przeglądarka	Wsparcie (od wersji)
Google Chrome	12.0
Internet Explorer	brak
Mozilla Firefox	brak
Safari	brak
Android Browser	brak
Chrome for Android	35.0
iOS Safari	brak

Tablica 2.4: Wsparcie mechanizmu Filesystem API w przeglądarkach internetowych.

się do Google Chrome oraz Chrome for Android. W związku z niskim zainteresowaniem innych firm zajmujących się rozwijaniem przeglądarek internetowych, organizacja W3C zawiesiła prace nad tym standardem.

Filesystem API jest bardzo dobrym rozwiązaniem dla aplikacji, które wymagają przechowywania dużej ilości zasobów - plików audio, wideo, zdjęć, dokumentów. W przypadku aplikacji które przechowują dane strukturalne, metody WebSQL oraz IndexedDB okażą się dużo bardziej efektywne. Ze względu na niskie wsparcie przeglądarek oraz zawieszenie prac nad rozwojem specyfikacji Filesystem, mechanizm ten nie spełnia wymagań stawianych aplikacji www wspierającej tryb offline.

2.3. Aplikacje webowe trybu offline

Tryb offline (w wolnym tłumaczeniu: "bez dostępu do sieci Internet") nie był dotychczas kojarzony z przeglądarkami internetowymi - programami z definicji służącymi do przeglądania zasobów WWW (*World Wide Web*). Aplikacje pracujące w tym trybie mogły być wywoływane jedynie poprzez ścieżkę rozpoczynającą się od **file://** i odwoływać się do danych znajdujących się na dysku twardym, flash lub płycie CD/DVD. Przykładem mogą być prezentacje korzystające z przeglądarki celem uruchomienia animacji napisanych za pomocą języka JavaScript.

Tryb offline pojawia się obecnie również w tak zwanych **Single-page applications**. Ich działanie polega na wykorzystaniu funkcji przeglądarki internetowej do interakcji z użytkownikiem i zapisania danych lokalnie. Przykład stanowić może TiddlyWiki - osobiste wiki umożliwiające tworzenie i edytowanie treści poprzez uruchomienie pojedynczej strony w przeglądarce bez dostępu do sieci Internet. Cel ten może zostać osiągnięty dzięki dobrodziejstwom oferowanym przez specyfikację HTML5.

Dochodzimy wreszcie do aplikacji wspierających zarówno tryb offline jak i online. Koncepcja ta wykorzystana jest w niniejszej pracy.

Często zasadność tworzenia aplikacji działającej bez dostępu do sieci jest poddawana w wątpliwość. Ogólnodostępność Internetu sprawia, że nawiązanie połączenia nie stanowi obecnie żadnej trudności.

Pomimo prawdziwości powyższego stwierdzenia należy brać pod uwagę względy oszczędnościowe i lokalizacyjne. Mobilność a co za tym idzie ciągła synchronizacja aplikacji zainstalowanych na urzą-

dzeniach przenośnych powoduje wzrost zużycia energii. Ponadto nadal istnieją miejsca, gdzie dostęp do sieci Internet jest zabroniony (pokład samolotu) znacząco utrudniony lub nawet niemożliwy.

Poniżej przedstawimy główne prawidłowości działania trybu offline przez pryzmat przeglądarek internetowych.

2.3.1. Pamięć cache

Pamięć podręczna przeglądarki (ang. *cache*) obejmuje miejsce na dysku twardym przeznaczone do przechowywania odwiedzonych stron WWW lub ich fragmentów. Najczęściej dotyczy to arkuszy stylów CSS, grafik a także skryptów w języku JavaScript. Wyróżniamy dwa główne powody stosowania cache-owania:

– **redukcja czasu dostępu do zasobów:** pamięć cache jest znacznie "bliżej" klienta, niż serwer oryginalny, dzięki czemu w krótszym czasie uzyskiwany jest dostęp do konkretnego zasobu i jego prezentacja,

– **redukcja ruchu sieciowego:** przechowywane w pamięci podręcznej zasoby mogą zostać użyte ponownie, co zmniejsza rozmiar pasma niezbędnego do uzyskania dostępu i oszczędza koszty ponoszone przez użytkownika.

Pamięć cache dokonuje aktualizacji przechowywanych treści najczęściej raz na sesję (za rozpoczęcie sesji przyjmujemy uruchomienie przeglądarki) tak, by użytkownik był na bieżąco ze zmianami treści. Poniżej przedstawiono ogólne zasady działania Web Cache:

1. Działaniem cache sterują nagłówki HTTP. Użytkownik może zdecydować między innymi o czasie przechowywania"świeżej" porcji danych w pamięci, ich dostępności (private/public) lub nawet o wyłączeniu cache-owania danych.
2. Żądania uwierzytelnienia lub żądania zabezpieczone (np. HTTPS) nie są cache-owane.
3. Reprezentacja danych w cache jest uważana za "świeżą"(nie musi być sprawdzana z wersją danych na serwerze oryginalnym), jeśli ustawiono nagłówek HTTP odpowiadający za czas życia danych lub jeśli dane były cache-owane niedawno.
4. Jeśli dane są nieaktualne do serwera zostanie wysłane zapytanie o walidację.
5. W sytuacji utraty połączenia internetowego pamięć cache może wysyłać dane bez weryfikacji ich z danymi pochodzącymi z oryginalnego serwera.

W przypadku braku dostępu do nagłówków HTTP możliwa jest edycja ustawień pamięci cache za pomocą znaczników **meta** zawartych w sekcji head definicji dokumentu HTML.

Proces zapisywania w pamięci podręcznej jest w dużej mierze zautomatyzowany i związany z wyświetleniem witryny. Należy rozgraniczyć na czym polegać, w kontekście projektu OffCalendar, będzie przechowywanie lokalne.

Użytkownik manualnie wywoła proces zapisu danych wykonując akcje na interfejsie aplikacji jak na przykład dodanie/edycja/usunięcie wydarzenia, edycja ustawień etc. Bez ingerencji dane nie trafią do pamięci podręcznej w odróżnieniu od cache-owania wykonywanego automatycznie podczas wyświetlenia dokumentu HTML.

2.3.2. Detekcja połączenia sieciowego

Wymogiem działania aplikacji jest automatyczna detekcja połączenia internetowego i przejście do trybu online/offline. Jest to spowodowane koniecznością uruchomienia zapisu lokalnego lub synchronizacji danych do serwera aplikacji.

Specyfikacja HTML5 oferuje różne mechanizmy wykrywania połączenia sieciowego. Różnią się one poziomem wsparcia w przeglądarkach.

Jednym z nich jest skorzystanie z tzw. Obserwatora Zdarzeń (*EventListener*). HTML5 udostępnia obserwatory **online** oraz **offline**.

Połączenie internetowe może także zostać zweryfikowane za pomocą obiektu **XMLHttpRequest (XHR)**. Jest to obiekt języków skryptowych (jak np. JavaScript) umożliwiający wykonanie żądań po załadowaniu się dokumentu HTML. Są one realizowane asynchronicznie (w tle) i nie przerywają interakcji użytkownika ze stroną. Badanie dostępu do Internetu ogranicza się do sprawdzenia odpowiedzi z serwera.

Biblioteką również wykorzystującą technologię AJAX (*Asynchronous JavaScript and XML*) w detekcji połączenia jest Offline.js. Umożliwia ona wykrycie utraconego połączenia oraz ponowne wykonanie żądań, które w wyniku owej utraty nie doszły do skutku. Dodatkowo oferuje szablony służące do wyświetlenia odpowiednich informacji dla użytkownika.

Szczegóły zastosowanych mechanizmów zostaną omówione w rozdziałach poświęconych projektowi i implementacji systemu.

2.4. Replikacja i synchronizacja danych

Niezbędnym elementem aplikacji www wspierającej tryb offline jest mechanizm przechowywania danych po stronie klienta. W momencie zerwania połączenia sieciowego i utraty komunikacji z serwerem, użytkownik powinien posiadać lokalną replikę danych znajdujących się na serwerze. W replice tej powinny zostać zapisane wszystkie wykonane przez niego akcje.

W chwili uzyskania ponownego dostępu do sieci, dane przechowywane w lokalnej bazie danych muszą zostać zsynchronizowane z danymi znajdującymi się na serwerze aby możliwa była ich dalsza propagacja do pozostałych urządzeń z których korzysta dany użytkownik.

Aby aplikacja była w stanie wykonać powyższe zadania, konieczne jest dobór odpowiedniej metody replikacji oraz sychronizacji danych, która pozwoli na stabilne i spójne działanie aplikacji, niezależnie od akcji wykonywanych przez użytkownika oraz stanu połączenia sieciowego.

2.4.1. Podstawowe pojęcia

Replikacja danych jest procesem w którym operacje wykonane na danej instacji bazy danych są powielane na pozostałych instancjach objętych mechanizmem replikacji. Głównym celem replikacji jest zwiększenie dostępności oraz szybkości usług sieciowych. Dostępność jest zwiększona poprzez możliowść dostępu do danych w przypadku gdy dostępna jest tylko część replik. Szybkość usług jest zwiększona poprzez dostęp użytkowników do najbliższej, względem ich lokacji, repliki. Prowadzi to do mniejszego opóźnienia sieciowego oraz rozłożenia obciążenia pomiędzy poszczególne repliki wchodzące w skład systemu.

W przypadku aplikacji www wspierających tryb offline mamy do czynienia z przypadkiem, gdy mechanizm lokalnego przechowywania danych obsługiwany przez przeglądarkę użytkownika staje się jedną z replik danych.

Transakcją nazywamy zbiór operacji na bazie danych, które tworzą pewną całość. Aby zmiany wprowadzone przez transakcję zostały zapisane w bazie danych, wszystkie operacje składające się na transakcję muszą zostać poprawnie wykonane. Transakcje powinny spełniać warunki opisane w zasadach ACID (Atomowość, Spójność, Izolacja, Trwałość):

- Atomowość - w przypadku niepowodzenia przynajmniej jedenej operacji wchodzącej w skład transakcji cała transakcja jest anulowana, pozostawiając dane w niezmienionej formie,

- Spójność - przed, w trakcie oraz po wykonaniu transakcji system pozostaje w spójnym stanie i nie naruszona zasady integralności danych,

- Izolacja - w przypadku wielu transakcji wykonywanych współbieżnie, zmiany dokonywane przez poszczególne transakcje są widoczne tylko i wyłącznie z perspektywy danej transakcji.

- Trwałość - w przypadku awarii, dane wszystkich transakcji które zostały zatwierdzone są zapisane w systemie i będą dostępne po ponownym jego uruchomieniu.

Szeregowalność jest pożądaną własnością systemów bazodanowych, która umożliwia zarządzanie operacjami wykonywanymi współbieżnie na bazie danych w taki sposób, aby wyeliminować niepożądane interakcje między nimi. Szeregowalność zapewnia taki sam stan końcowy danych po współbieżnym wykonaniu danych transakcji, jak i po wykonaniu tych transakcji szeregowo.

Zakleszczenie jest zdarzeniem, które zachodzi w sytuacji gdy dwa procesy, każdy z nich wykonujący transakcję, modyfikują wspólny zbiór rekordów w odwrotnej kolejności Prowadzi to do sytuacji, w której każdy z procesów czeka na zakończenie przeciwnego, co w przypadku zakleszczenia nie następuje. System zarządzania bazą danych odpowiada za wykrywanie zakleszczeń i ich rozwiązywanie, zazwyczaj przez anulowanie jednej z transakcji.

2.4.2. Klasyfikacja technik replikacji

Istnieje wiele technik replikacji które pozwalają na skuteczną oraz stabilną pracę rozproszonych systemów bazodanowych. Główne cechy replikacji które muszą zostać określone to:

– zakres replikowanych danych,

– sposób sychronizacji pomiędzy węzłami,

– poziom dostępu do danych w zależności od statusu repliki.

Poniżej przedstawione zostały klasyfikacje które pomagą określić typ replikacji, który oferuje najlepszą funkcjonalność w przypadku aplikacji www wspierających tryb offline.

Replikacja całościowa - Replikacja częściowa

Replikacja całościowa zapewnia dostępność pełnej kopii danych na wszystkich węzłach wchodzących w skład systemu.

Replikacja częściowa pozwala na wybór danych, które będą replikowane, przy czym zbiór danych podlegających replikacji jest określany indywidualnie dla każdego węzła wchodzącego w skład systemu. Replikacja częściowa może dzielić zbiór danych na dwa sposoby:

– horyzontalnie - z danej tabeli wybrany jest zbiór wierszy zawierający wszystkie wartości dla kolumn zdefiniowanych w tej tabeli.

– wertykalnie - z danej dabeli wybrany jest zbiór kolumn, których wartości dla każdego wiersza są replikowane.

Replikacja synchroniczna - Replikacja asynchroniczna

Replikacja synchroniczna, tzw. "gorliwa", stara się utrzymać zsynchronizowane dane pomiędzy wszystkimi węzłami wchodzącymi w skład systemu. W przypadku replikacji synchronicznej, w skład transakcji wchodzi propagacja danych do wszystkich węzłów - w przypadku wystąpienia konfliktu transakcja jest anulowana. Zapewnia to ochronę przed możliwymi niespójnościami w bazie danych, jednak ryzyko zakleszczenia znacząco wzrasta wraz ze wzrostem liczby transakcji.

Replikacja asynchroniczna, tzn. łeniwa", pozwala na niezależne wykonywanie transakcji na każdym z węzłów. Po wykonaniu transakcji na jednym z węzłów, dane są asynchronicznie propagowane do pozostałych węzłów. Replikacja leniwa nie wymaga wysokiej dostępności węzłów, jednak może prowadzić do konfliktów.

Przykładem obrazującym różnicę pomiędzy replikacją leniwą a replikacją gorliwą może być próba wykonania dwóch przelewów z dwóch różnych węzłów objętych replikacją, gdzie obciążenie rachunku

przekracza dostępne saldo. W przypadku replikacji gorliwej, pierwsza z transakcji zostanie wykonana pomyślnie, druga zwróci komunikat błędu. W przypadku replikacji leniwej, obie transakcje zakończą się sukcesem, jednak w fazie synchronizacji danych dojdzie do konfliktu, który będzie musiał zostać obsłużony.

Replikacja pesymistyczna - Replikacja optymistyczna

Replikacja pesymistyczna zakłada, że stan obiektów przechowywanych w każdym węźle jest identyczny oraz aktualny. W celu uniknięcia konfliktów, w przypadku nieaktualnego stanu danych w węźle, dostęp do danych jest blokowany.

Replikacja optymistyczna nie ogranicza w żaden sposób dostępu do danych, niezależnie od stanu w którym znajduje się replika. Każda z replik przechowuje historię wykonanych transakcji. Ze względu na niezależne wykonywanie transakcji na każdym węźle, w momencie synchronizacji system jest narażony na możliwe konflikty danych. Prowadzi to do konieczności modyfikacji części transakcji i ponownego ich wykonania, co może skutkować innym stanem bazy danych po zakończeniu synchronizacji.

2.4.3. Replikacja danych w aplikacji www wspierającej tryb offline

Aplikacja www wspierająca tryb offline powinna minimalizować ilość danych przesyłanych pomiędzy poszczególnymi węzłami systemu. Ze względu na prywatność danych, lokalna replika powinna posiadać wyłącznie dane powiązane z konkretnym, zalogowanym w danym momencie do aplikacji, użytkownikiem. Dodatkowo, uwzględniając coraz większą aktywność użytkowników korzystających z urządzeń mobilnych, aplikacja powinna ograniczyć ilość przesyłanych danych do minimum. Z tego względu, replikacja częściowa jest w tym przypadku znacznie lepszym wyborem od replikacji całościowej.

Z uwagi na wymagania aplikacji www oraz nacisk położony na pełną jej funkcjonalność w trybie offline, synchronizacja danych pomiędzy węzłami powinna odbywać się w sposób łeniwy". Przechowywanie danych w lokalnej replice oraz późniejsza asynchroniczna wymiana informacji pomiędzy węzłami powinna zapewnić bardzo płynną interakcję z interfejsem użytkownika, co jest jednym z wymagań aplikacji. W przypadku użytkowników posiadających repliki danych na wielu urządzeniach które przez większość czasu są fizycznie wyłączone lub nie posiadają dostępu do sieci, zastosowanie replikacji "gorliwejńie byłoby możliwe.

Użytkownik korzystający z aplikacji na urządzeniu mobilnym może znaleźć się w rejonie o słabym zasięgu sieci telefonicznej. W celu oszczędności energii oraz minimalizacji transferu danych przez sieć mobilną, użytkownik może wyłączyć mobilny transfer danych. W przypadku zastosowania replikacji pesymistycznej, uniemożliwiłoby to komunikację z lokalną repliką danych ze względu na nieosiągalność pozostałych węzłów. Z tego względu w aplikacji www wspierającej tryb offline powinna zostać zastosowana replikacja optymistyczna.

Zastosowanie replikacji posiadającej powyższe cechy może doprowadzić do sytuacji, w których podczas procesu synchronizacji wystąpi konflikt transakcji. Aplikacja powinna posiadać wewnętrzne algorytmy które obsłużą takie sytuacje, zaś w przypadku braku możliwości programatycznej obsługi, użytkownik powinien zostać poinformowany o konflikcie oraz o możliwościach jego rozwiązania.

2.5. Analiza wymagań

OffCalendar jest systemem do zarządzania wydarzeniami użytkownika. Zgodnie z nazwą zorganizowany jest w formie interaktywnego kalendarza. Główną zaletą aplikacji jest pełne wsparcie zarówno trybu online jak i offline co uniezależnia ją od stałego dostępu do sieci Internet. Poniższa analiza sugeruje jedynie możliwe zastosowanie systemu zgodnie z jego pierwotnymi założeniami, aczkolwiek nie wyklucza innej aktywności. Szczegółowy opis możliwości systemu zawarty jest w treści poniższych podrozdziałów.

2.5.1. Wymagania funkcjonalne

Zadaniem aplikacji OffCalendar jest dodawanie, edycja oraz usuwanie wydarzeń. Przez wydarzenie rozumiemy notatkę składającą się z nazwy, czasu trwania oraz treści. Dzięki zaimplementowanemu systemowi autentykacji każdy użytkownik posiada konto, co jednoznacznie identyfikuje wprowadzone przez niego dane. Ponadto użytkownik ma do dyspozycji system powiadomień o nadchodzących wydarzeniach. Jest on w pełni konfigurowalny.

Podane akcje mogą być wykonywane zarówno w trybie offline jak i przy aktywnym połączeniu z siecią Internet. Dane gromadzone lokalnie zostają zsynchronizowane z tymi obecnymi w bazie danych serwera aplikacji w momencie wykrycia połączenia. Sam proces synchronizacji może zakończyć się niepowodzeniem, co zostanie obsłużone automatycznie przez zaimplementowane algorytmy lub manualnie, po uprzednim poinformowaniu użytkownika o problemie.

Krytycznym wymaganiem funkcjonalnym systemu jest zapewnienie integralności danych i zapewnienie ciągłości transakcji pomimo problemów z połączeniem sieciowym. Przejście pomiędzy trybami online/offline powinno być płynne i nie wpływać negatywnie na wydajność interfejsu użytkownika.

Z uwagi na zastosowanie optymistycznej replikacji bazy danych należy zwrócić baczną uwagę na możliwość wystąpienia konfliktów pomiędzy żądaniami i ich obsługę.

2.5.2. Wymagania niefunkcjonalne

Projektowany system składa się z dwóch zasadniczych części: aplikacji klienckiej stworzonej w technologii HTML5 oraz części serwerowej złożonej z serwera www oraz bazy danych. Poszczególne komponenty stawiają określone wymagania z uwagi na specyfikę projektu.

Część serwerowa została zaimplementowana w języku PHP (*PHP Hypertext Preprocessor*) w wersji 5.5 z wykorzystaniem frameworka MVC (*Model View Controller*) CodeIgniter. Relacyjna baza danych wykorzystana w projekcie to MySQL.

Aplikacja kliencka z założenia dedykowana jest przeglądarkom internetowym. Interfejs programu zbudowany został w oparciu o język znaczników HTML5. Za warstwę wizualną odpowiada język kaskadowych arkuszy stylów CSS. Asynchroniczna obsługa systemu możliwa jest dzięki zastosowaniu języka skryptowego JavaScript. Wspomniane rozwiązania umożliwiają stworzenie przejrzystego i responsywnego interfejsu przystosowanego zarówno do przeglądarek desktopowych jak i mobilnych.

Wymogiem dla technologii lokalnego przechowywania danych wprowadzanych przez użytkownika jest możliwość przechowywania stosunkowo złożonych rekordów bez konieczności ich serializacji, szerokie wsparcie wśród przeglądarek oraz szybkie wyszukiwanie. IndexedDB łączy wszystkie wymienione wymagania i stanowi najlepszy wybór wśród obecnie stosowanych technologii. Dodatkowym atutem jest bogata i aktualizowana dokumentacja.

Komunikacja pomiędzy częściami systemu opiera się na architekturze REST. Wybór podyktowany został niewielkim narzutem obliczeniowym dzięki zastosowaniu lekkiego formatu JSON (*JavaScript Object Notation*) oraz braku konieczności opakowywania komunikatów (jak w przypadku protokołu SOAP). Wystarczy wyspecyfikować adres zasobu oraz metodę dostępu (GET, POST, PUT) a także wersję protkołu HTTP.

Wyselekcjonowany zestaw technologii i języków programowania ma na celu zapewnienie bezpieczeństwa, zwiększenie szybkości aplikacji i minimalizację zasobów potrzebnych do komunikacji między klientem a serwerem. Zastosowanie frameworków tj. CodeIgniter oraz jQuery wpływa z kolei na skrócenie czasu implementacji do niezbędnego minimum.

2.6. Wykorzystane technologie i narzędzia

Aby aplikacja www wspierająca tryb offline mogła zostać wykonana, konieczny był dobór odpowiednich narzędzi programistycznych oraz technologii umożliwiających zrealizowanie założonego celu.

Najważniejsze elementy które muszą zostać wykonane w części klienckiej aplikacji to szata graficzna, napisanie kodu HTML tworzącego strukturę interfejsu oraz dodanie odpowiednich reguł w ka-

skadowych arkuszach stylów tworzących warstwę prezentacji. Aby umożliwić pracę aplikacji w trybie offline, konieczna jest integracja wybranego mechanizmu lokalnego przechowywania danych oraz stworzenie komponentu odpowiedzialnego za synchronizację danych przy użyciu języka JavaScript.

W części serwerowej aplikacji, głównymi zadaniem będzie stworzenie struktury bazy danych MySQL przechowującej dane użytkowników, stworzenie usług odpowiadających za komunikację z częścią kliencką aplikacji oraz komponentu odpowiedzialnego za rozwiązywanie konfliktów mogących powstać przy procesie synchronizacji danych.

2.6.1. Netbeans IDE

Netbeans IDE jest zintegrowanym środowiskiem programistycznym posiadającym bardzo dobre wsparcie zarówno dla używanych przez nas technologii serwerowych (PHP, MySQL, Apache), webowych (HTML, CSS, JavaScript) jak i użytego w projekcie systemu kontroli wersji (GIT). Dodatkowym atutem Netbeans IDE jest jego udostępnianie na licencji GNU General Public License, co czyni oprogramowanie darmowym zarówno w użytku prywatnym jak i komercyjnym.

Alternatywą dla Netbeans IDE jest PHPStorm – doskonałe środowisko rozwijane przez firmę JetBrains. Posiada ono wszystkie jego zalety, a ponadto oferuje szereg zaawansowanych narzędzi służących do połączenia ze zdalnym serwerem, testowania oraz debugowania rozwijanej aplikacji. PHPStorm nie został wykorzystany w projekcie, ponieważ jest to płatne oprogramowanie, którego dodatkowe narzędzia nie znajdą zastosowania podczas procesu tworzenia aplikacji www wspierającej tryb offline.

2.6.2. Adobe Photoshop CS3

Każda aplikacja www powinno posiadać szatę graficzną dostosowaną do jej potrzeb. Adobe Photoshop jest najpopularniejszym na rynku narzędziem do edycji grafiki rastrowej oraz tworzenia interfejsów strow www. Do największych zalet należy zaliczyć ergonomiczny interfejs, zaawansowane możliwości operacji na warstwach oraz bogaty zestaw efektów, który możemy nadać danemu elementowi. Szybka możliwość eksportu poszczególnych grafik, osadzanych później w kodzie HTML, jest kolejną funkcjonalnością która sprawia że program spełnia wszystkie wymagania.

Alternatywnym oprogramowaniem umożliwiającym edycję grafiki rastrowej jest GIMP (*GNU Image Manipulation Program*). GIMP jest darmowym odpowiednikiem Adobe Photoshop, jednak oferującym znacznie bardziej ograniczone możliwości manipulacji warstwami, tworzenia szablonów oraz nadawania efektów poszczególnym elementom. Niska ergonomia oraz intuicyjność interfejsu sprawia, że czas potrzebny na uzyskanie takiego samego efektu końcowego jak w Adobe Photoshop jest znacznie dłuższy. Z tego względu, mimo darmowej licencji, oprogramowanie GIMP nie zostało wykorzystane w projekcie.

2.6.3. HTML5

HTML jest podstawowym językiem służącym do tworzenia stron www. Zadaniem znaczników HTML jest nadanie semantycznego znaczenia elementom renderowanym na stronie. Inne zastosowania znaczników obejmują m. in. za załączenie odpowiednich arkuszy stylów, wczytanie bibliotek i skryptów języka JavaScript, określenie sposobu indeksowania strony oraz reguły specyfikujące sposób przechowywania dokumentów w pamięci cache.

Z punktu widzenia aplikacji wspierającej tryb offline, szczególnie ważnymi elementami wywodzącymi się ze specyfikacji HTML5 są mechanizmy lokalnego przechowywania danych oraz instrukcje pozwalające określić metodę przechowywania poszczególnych zasobów strony www w pamięci cache. Dodatkowym atutem języka HTML5 jest natywna walidacja danych wprowadzanych do formularzy przez użytkownika oraz rozszerzony zbiór semantycznych znaczników, pozwalający na trafniejsze opisanie elementów strukturalnych dokumentu, takich jak nagłówek, stopka, nawigacja czy panel boczny.

2.6.4. Kaskadowe arkusze stylów

Kaskadowe arkusze stylów (*CSS - Cascading Style Sheets*) to nieodłączny element stron www. Kaskadowe arkusze stylów zostały zaprojektowane z myślą o separacji warstwy prezentacji strony od jej treści. Poprzez zawarcie reguł opisujących sposób wyświetlania poszczególnych elementów dokumentu HTML w zewnętrznym arkuszu stylów, możliwe jest łatwe ich dołączenie do wszystkich dokumentów. Edycja danej reguły w zewnętrznym pliku CSS powoduje aktualizację wszystkich stron używających danego pliku, co znacznie usprawnia pracę nad większymi projektami.

Reguły w kaskadowych arkuszach stylów mogą zostać załączane warunkowo. Typowym zastosowaniem są tutaj odrębne style dla drukowanych dokumentów. Bardziej zaawansowane reguły pozwalają na zastosowanie odrębnych zestawów reguł w zależności od rozdzielczości ekranu czy też rozmiaru okna przeglądarki. Umożliwia to tworzenie responsywnych interfejsów, zachowujących ergonomię niezależnie od urządzenia na jakim interfejs jest wyświetlony.

2.6.5. JavaScript - skryptowy język programowania

Język JavaScript jest nieodłącznym elementem interaktywnych aplikacji webowych. Szczególnie dużą rolę pełni on w aplikacjach wspierających tryb offline, jako że jest to jedyna technologia która pozwala zastąpić dynamiczną treść generowaną na zdalnym serwerze. Dzieje się to poprzez monitorowanie akcji wykonywanych przez użytkownika aplikacji oraz ich obsługę po stronie przeglądarki internetowej.

Mechanizmy lokalnego przechowywania danych są silnie uzależnione od języka JavaScript. Wszystkie operacje zapisu, odczytu oraz modyfikacji danych obdywają się za jego pośrednictwem, poprzez

odwołanie się do obiektów oraz metod wyszczególniowych z specyfikacji danej metody lokalnego przechowywania danych.

W ostatnich latach nastąpił znaczny rozwój frameworków oraz bibliotek języka JavaScript. Udostępniane przez nie narzędzia znacznie ułatwiają pracę nad tworzeniem dynamicznych elementów aplikacji www.

Wśród najpopularniejszych bibliotek ułatwiających manipulację drzewem dokumentu DOM (*Document Object Model*) znajdują się jQuery, Prototype, MooTools. Ze względu na bardzo dobrą dokumentację oraz możliwość łatwego rozszerzenia natywnej funkcjonalności poprzez bogaty zestaw wtyczek, biblioteka jQuery, wraz z dodatkiem jQuery UI, zostanie wykorzystana w projekcie web aplikacji wspierającej tryb offline.

Złożone aplikacje webowe potrzebują bibliotek oferujących rozwiązania tworzące szkielet architektury, która pozwala na logiczny podział modułów aplikacji oraz funkcjonalności przez nią realizowaną. Do najpopularniejszych bibliotek zaliczymy tutaj AngularJS, Cappuccino czy Knockout. Ze względu na specyfikę aplikacji wspierającej tryb offline oraz konieczność stworzenia dedykowanej warstwy odpowiedzialnej za przechowywania oraz synchronizację danych, biblioteki tworzące architekturę aplikacji nie zostaną zastosowane.

2.6.6. Apache 2.4.9

Aby aplikacja www była osiągalna dla użytkowników, konieczne jest pośrednictwo serwera www. Apache HTTP Server jest obecnie najpopularniejszym serwerem obsługującym aplikacje webowe. Jest udostępniany na licencji open-source oraz posiada wsparcie zarówno na systemach Unix jak i Windows.

Ze względu na udostępniony zestaw popularnych konfiguracji oraz dużą ilość przewodników opisujących poszczególne funkcjonalności serwera, proces instalacji oraz konfiguracji jest skrócony do minimum. Szczególnie ważne jest to przy niewielkich projektach, w których czas konfiguracji środowiska programistycznego, lokalnego oraz zdalnego serwera, nie powinien być jednym z elementów pochłaniających najwięcej czasu.

2.6.7. PHP 5.5

PHP to skryptowy język programowania zaprojektowany pod kątem użycia w aplikacjach webowych. Kod języka PHP nie jest kompilowany, zamiast tego jest on prasowany oraz interpretowany przez interpreter PHP.

Ze względu na przeniesienie ciężaru logiki aplikacji na stronę klienta (JavaScript) technologia serwerowa powinna być możliwie prosta w zastosowaniu, przy czym posiadająca wsparcie dla wszystkich

wymaganych funkcjonalności. Część serwerowa aplikacji będzie ograniczała się do podstawowych widoków, warstwy dostępu do bazy danych oraz webservice-ów odpowiedzialnych za uwierzytelnienie użytkownika czy też synchronizację danych.

Język PHP oferuje bardzo dobre wsparcie powyższych elementów aplikacji przy jednoczesnej minimalizacji narzutu technologicznego. PHP, w połączeniu z serwerem Apache oraz technologią MySQL stanowi spójny zestaw technologii pozwalający na bardzo dobrą skalowalność aplikacji, co jest szczególnie ważne w przypadku późniejszej rozbudowy projektu.

2.6.8. MySQL 5.6.17

Jednym z kluczowych elementów aplikacji www wspierającej tryb offline jest sposób przechowywania danych po stronie klienta oraz późniejsza ich synchronizacja. Ze względu strukturę danych oraz wymaganą transakcyjność przy procesie replikacji danych, zastosowanie relacyjnej bazy danych jest tutaj najlepszym rozwiązaniem. Zarówno MySQL jak i PostgreSQL są zaawansowanymi, darmowymi systemami bazodanowymi oferującymi wymaganą funkcjonalność.

Technologia MySQL posiada silnik InnoDB, oferujący bardzo wysoką wydajność w przypadku aktualizacji rekordów znajdujących się w bazie danych. Poprzez zastosowanie blokady na poziomie wierszy, poszczególne procesy dokonujące synchronizacji danych pomiędzy węzłami systemu powinny zapewniać dobrą wydajność, niezależnie od liczby współbieżnych operacji. MySQL posiada również zaawansowany system replikacji, co jest szczególnie ważne w przypadku problemów z przepustowością pojedynczej instancji bazy danych oraz wyczerpaniu możliwości skalowania wertykalnego. Powyższe cechy sprawiają, że system MySQL zostanie zastosowany w budowie aplikacji www wspierającej tryb offline.