

Разработка публичного веб-сервиса интерактивной карты на основе Yandex maps API

Андриенко Иван Сергеевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Пащенко Екатерина Александровна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается реализация публичного веб-сервиса интерактивной карты кладбища, предназначенного для поиска и визуализации мест захоронений. Описываются архитектурные решения клиентской части, способы интеграции картографического API, механизмы ограничения области отображения, динамической подгрузки данных и взаимодействия пользователя с объектами карты. Показано, что использование современных веб-технологий позволяет реализовать наглядный и удобный сервис навигации по территории кладбища без применения специализированного программного обеспечения.

Ключевые слова: веб-картография, интерактивная карта, Yandex Maps API, Next.js, геоинформационные системы, визуализация данных, поиск объектов.

Development of a public interactive map web service based on the Yandex maps API

Andrienko Ivan Sergeevich

Sholom-Aleichem Priamursky State University

Student

Paschenko Ekaterina Alexandrovna

Sholom-Aleichem Priamursky State University

Student

Abstract

The article discusses the implementation of a public web service for interactive cemetery maps designed to search and visualize burial sites. The architectural solutions of the client side, methods of integrating the cartographic API, mechanisms for limiting the display area, dynamically loading data, and user interaction with map objects are described. It is shown that the use of modern web technologies makes it possible to implement a visual and convenient service for navigating the cemetery territory without using specialized software.

Keywords: web cartography, interactive map, Yandex Maps API, Next.js, geographic information systems, data visualization, object search.

1 Введение

1.1 Актуальность

Актуальность исследования обусловлена необходимостью цифровизации навигационных и справочных сервисов, связанных с общественно значимыми объектами. Кладбища представляют собой территориально сложные пространства, где традиционные способы поиска мест захоронений являются малоэффективными. Использование веб-картографических сервисов позволяет повысить доступность информации, упростить поиск объектов и обеспечить удобное взаимодействие пользователей с пространственными данными в браузере.

1.2 Обзор исследований

Интеграция картографических сервисов в программные приложения на основе React и TypeScript находит отражение в современной научной литературе. В работе А.А. Севера рассматривается использование API карт Google Maps, Yandex.Maps и 2GIS в приложении на JavaFX, где анализируются особенности подключения различных картографических сервисов, их программные интерфейсы и практические аспекты реализации отображения карт и маркеров в десктопной среде [1]. Н.И. Филиппов и А.В.Лошаков выполняют сравнительный анализ сервисов Google Maps и Yandex.Maps, оценивая их функциональные возможности, точность картографических данных, удобство интеграции и особенности использования в прикладных задачах [2]. В статье Д.А. Хайруллина и М.Р. Богданова рассматриваются особенности применения статической типизации в React-приложениях, включая методики типизации компонентов, хуков и состояния, а также влияние строгой типизации на надёжность пользовательских интерфейсов [3]. К.Р. Удальцов исследует вопросы обеспечения качества кода в командной разработке, предлагая формализацию шаблонов безопасных pull request-ов для проектов на React и TypeScript в рамках Git Flow [4]. Особое внимание созданию библиотек компонентов на основе TypeScript уделяется в работе Т.М. Воронко, где описываются принципы модульности, повторного использования кода и проектирования архитектуры компонентов для масштабируемых React-решений [5].

1.3 Цель исследования

Цель исследования – разработать и описать публичный веб-сервис интерактивной карты кладбища, обеспечивающий визуализацию территории, поиск мест захоронений и отображение детальной информации с использованием картографического API.

2 Материалы и методы

В качестве программной основы использовался фронтенд-фреймворк Next.js с применением App Router. Для визуализации пространственных данных применялся JavaScript API Яндекс.Карт версии 2.1. В работе использовались методы клиентской обработки геоданных, динамической загрузки внешних скриптов, ограничения области карты, а также асинхронные HTTP-запросы для получения точек захоронений по видимой области и поисковому запросу.

3 Результаты и обсуждения

Для корректной работы сервиса в среде Next.js с серверным рендерингом была реализована динамическая загрузка JavaScript API Яндекс.Карт версии 2.1. Подключение API выполняется исключительно на клиентской стороне, что позволяет избежать ошибок, связанных с отсутствием объекта window при серверном рендеринге.

Реализация основана на проверке наличия загруженного API и асинхронной подгрузке скрипта в случае его отсутствия. Такой подход обеспечивает однократную инициализацию картографической библиотеки и предотвращает повторное подключение при повторных рендерах компонента (рис. 1).

```
75
76 const loadYmaps = () => {
77   if (typeof window === 'undefined') return Promise.reject('no window');
78   if (window.ymaps) return Promise.resolve(window.ymaps);
79   if (ymapsLoader) return ymapsLoader;
80
81   ymapsLoader = new Promise<any>((resolve, reject) => {
82     const existing = document.getElementById('ymaps-script') as HTMLScriptElement | null;
83     if (existing) {
84       existing.addEventListener('load', () => resolve(window.ymaps));
85       existing.addEventListener('error', () => reject(new Error('Не удалось загрузить Яндекс.Карты 2.1')));
86       return;
87     }
88     const script = document.createElement('script');
89     script.id = 'ymaps-script';
90     script.src = `https://api-maps.yandex.ru/2.1/?lang=ru_RU&apikey=${YM_KEY}`;
91     script.async = true;
92     script.onload = () => resolve(window.ymaps);
93     script.onerror = () => {
94       ymapsLoader = null;
95       reject(new Error('Не удалось загрузить Яндекс.Карты 2.1'));
96     };
97     document.head.appendChild(script);
98   });
99
100   return ymapsLoader;
101 };
102
```

Рис. 1. Динамическая загрузка Yandex Maps API 2.1

После загрузки картографического API выполняется инициализация объекта карты. Карта создаётся один раз и привязывается к DOM-элементу страницы. На этапе инициализации задаются параметры масштабирования, список доступных элементов управления и ограничения на минимальный и максимальный уровень зума.

Дополнительно отключается интерактивность стандартных ROI-объектов и удаляются неиспользуемые элементы управления, что снижает визуальный шум и фокусирует внимание пользователя исключительно на

данных сервиса. Ограничение области карты предотвращает навигацию за пределы кладбища и повышает удобство работы с интерфейсом (рис. 2).

```
163     const initMap = async () => {
164       try {
165         const ymaps = await loadYmaps();
166         if (cancelled) return;
167         await ymaps.ready();
168
169         if (mapInstanceRef.current) {
170           mapInstanceRef.current.destroy();
171           mapInstanceRef.current = null;
172         }
173
174         const map = new ymaps.Map(mapRef.current, {
175           center: DEFAULT_CENTER,
176           zoom: 17,
177           minZoom: 15,
178           maxZoom: 19,
179           controls: ['zoomControl', 'typeSelector', 'geolocationControl'],
180           restrictMapArea: BOUNDS,
181           suppressMapOpenBlock: true,
182         });
183         map.options.set('restrictMapArea', BOUNDS);
184         map.options.set('yandexMapDisablePoiInteractivity', true);
185         map.options.set('suppressMapOpenBlock', true);
186         map.controls.remove('searchControl');
187         map.controls.remove('trafficControl');
188       }
```

Рис. 2. Инициализация карты и отключение стандартных элементов управления

Для ограничения области работы сервиса задаётся центр кладбища и вычисляется прямоугольная область допустимого отображения (bounding box). Границы области рассчитываются на основе координат центра с использованием фиксированных смещений по широте и долготы. Полученная область используется как для ограничения перемещения карты, так и для формирования запросов к API при загрузке точек захоронений. Такой подход гарантирует, что пользователь работает исключительно в пределах территории кладбища.

Для наглядного обозначения границ кладбища поверх карты накладываются полигональные объекты. Координаты полигонов подготавливаются заранее и приводятся к формату, используемому картографическим API (рис. 3).

```
189 // зафиксировать стартовые границы и не давать уходить за BOUNDS
190 map.setBounds(BOUNDS, { checkZoomRange: true });
191
192 mapInstanceRef.current = map;
193
194 // Показать местоположение пользователя (без центрирования по умолчанию)
195 ymaps.geolocation
196   .get({ provider: 'browser', mapStateAutoApply: false })
197   .then((res: any) => {
198     const coords = res?.geoObjects?.get(0)?.geometry?.getCoordinates();
199     if (coords && Array.isArray(coords)) {
200       if (userPlacemarkRef.current) {
201         map.geoObjects.remove(userPlacemarkRef.current);
202       }
203       const pm = new ymaps.Placemark(
204         coords,
205         { hintContent: 'Вы здесь' },
206         { preset: 'islands#blueCircleDotIcon' },
207       );
208       userPlacemarkRef.current = pm;
209       map.geoObjects.add(pm);
210     }
211   })
212   .catch(() => {});
213
214 // Полигоны
215 CEMETERY_POLYGONS_LATLON.forEach((coords) => {
216   const polygon = new ymaps.Polygon(
217     [coords],
218     {},
219     {
220       fillColor: 'rgba(176,112,60,0.14)',
221       strokeColor: palette.accent,
222       strokeOpacity: 0.65,
223       strokeWidth: 2,
224     },
225   );
```

Рис. 3. Задание центра и границ кладбища

Загрузка данных о захоронениях реализована на основе параметров текущей видимой области карты. При каждом завершении действия пользователя (перемещение или масштабирование карты) вычисляются актуальные границы области отображения, которые передаются в API-запрос в виде bounding box.

Каждая полученная запись о захоронении преобразуется в картографический маркер. Для маркеров задаются всплывающие подсказки и балуны с базовой информацией. При клике по маркеру выполняется обработка события, в ходе которой выбранный объект сохраняется в состоянии приложения и инициируется отображение детальной информации. Таким образом реализуется связка карты и пользовательского интерфейса без необходимости перезагрузки страницы (рис. 4).

```

324 const selectGrave = useCallback((grave: Grave) => {
325   const map = mapInstanceRef.current;
326   if (!map) return;
327   setSelectedId(grave.id);
328   setDetail(grave);
329   setPhotoIndex(0);
330   const [lon, lat] = grave.location.coordinates;
331   map.setCenter([lat, lon], 18, { duration: 300 });
332   const pm = placemarkMapRef.current[grave.id];
333   if (pm && pm.balloon) pm.balloon.open();
334 }, []);
335
336 // При клике на маркер в cluster: handled above
337
338 return (
339   <div
340     className="min-h-screen"
341     style={{ background: palette.ivory, color: palette.graphite, display: 'flex', flexDirection: 'column' }}
342   >
343     <header className="sticky top-0 z-20 border-b border-[#e1ddd5] bg-white/85 backdrop-blur">
344       <div className="mx-auto flex max-w-[1400px] flex-col gap-3 px-4 py-3 md:flex-row md:items-center md:justify-between">
345         <div className="flex items-center gap-3">
346           <div className="flex h-10 w-10 items-center justify-center rounded-lg bg-[#2d2d2d] text-white font-medium">
347             К
348           </div>
349           <div>
350             <p className="text-xs uppercase tracking-[0.08em]" style={{ color: palette.softGray }}>Городской проект</p>
351             <h1 className="text-lg font-semibold leading-tight">Карта памяти — захоронения</h1>
352           </div>
353         </div>
354       </div>
355       <div className="flex flex-wrap items-center gap-2">
356         <button
357           className="rounded-full border border-[#d5cec2] bg-white px-4 py-2 text-sm font-medium hover:border-[#d5cec2]
358         >

```

Рис. 4. Создание маркеров и обработка клика

Детальный просмотр информации о захоронении реализован в виде выезжающей боковой панели. Панель отображается поверх карты и содержит расширенные сведения об объекте, включая текстовое описание и фотоматериалы.

Анимация появления панели улучшает пользовательский опыт и позволяет сохранить контекст карты при просмотре деталей, не прерывая навигацию (рис. 5).

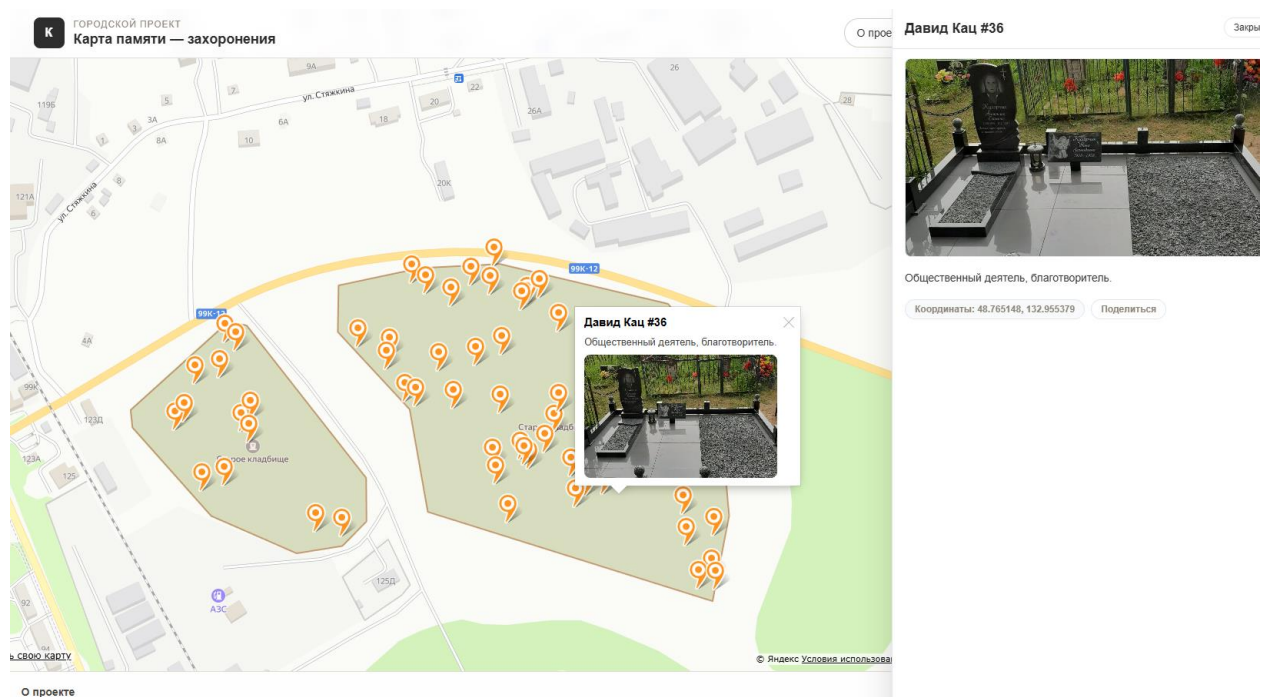


Рис. 5. Главный экран и правая панель детального просмотра

Реализованный подход показал, что использование динамической подгрузки картографического API, ограничение области отображения и загрузка данных по видимой области позволяют создать производительный и масштабируемый публичный сервис. Интеграция карты со списком объектов и детальной панелью обеспечивает интуитивное взаимодействие пользователя с пространственными данными.

4 Выводы

В результате работы реализован публичный веб-сервис интерактивной карты кладбища, основанный на современных веб-технологиях и картографическом API. Разработаны механизмы динамической загрузки картографической библиотеки, ограничения области отображения, визуализации территории, подгрузки и отображения точек захоронений, а также детального просмотра информации.

Проведённая реализация подтверждает возможность создания функциональных и удобных веб-картографических сервисов для общественно значимых задач без использования специализированного программного обеспечения и с минимальными требованиями к пользовательской стороне.

Библиографический список

1. Север А.А. Использование API карт Google Maps, Yandex.Maps и 2GIS в приложении JavaFX // Научные труды магистрантов и аспирантов. Нижневартонск, 2019. С. 265–268.
2. Филиппов Н.И., Лошаков А.В. Сравнительный анализ картографических сервисов Google Maps и Yandex Maps // Актуальные научные исследования в современном мире. 2021. № 4-3 (72). С. 128–133.
3. Хайруллин Д.А., Богданов М.Р. Изучение использования TypeScript в React-приложениях, преимущества статической типизации, подходы к типизации компонентов, хуков и состояния // Международный журнал гуманитарных и естественных наук. 2024. № 12-3 (99). С. 233–238.
4. Удальцов К.Р. Формализация шаблонов безопасных pull request-ов для React-приложений с TypeScript в командах с Git Flow // Актуальные проблемы социально-гуманитарного и научно-технического знания. 2025. № 4 (44). С. 35–36.
5. Воронко Т.М. Создание React-библиотеки с использованием TypeScript // Электронные системы и технологии. Материалы 58-й научной конференции аспирантов, магистрантов и студентов БГУИР. Минск, 2022. С. 200–202.