Sentry. Интеграция.

Для браузеров существует базовый пакет @sentry/browser и обёртка вокруг него @sentry/react для приложений, написанных с помощью ReactJS.

Документацию можно посмотреть на оф сайте: https://docs.sentry.io/platforms/javascript/guides/react/

Инициалиация

```
// index.js
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

import * as Sentry from "@sentry/react";

Sentry.init({
  dsn: "YOUR_SENTRY_DSN",
  integrations: [
    ...
  ],

});

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

После этого. sentry начнёт отсылать логи по указанному в поле`dsn` адресу для всех ошибок, который всплывают до `window.onerror` и `window.onunhandledrejecttion` обработчиков.

Так же мы можем указать среду в которой мы сейчас находимся при инициализации:

```
Sentry.init({
  ...,
  beforeSend(event) {
    event.environment = 'development';  // add or modify properties in the event
    return event;
  },
  ...
});
```

Мы так же можем логгировать ошибки самостоятельно в тех местах считаем нужным:

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => {
    // do something with data
  })
  .catch(error => {
    Sentry.captureException(error);
  });
```

В том числе настраивать дополнительные параметры scope-а:

```
Sentry.withScope(scope => {
  scope.setTag("my-tag", "my value");
  scope.setUser({ id: "user-id" });
  Sentry.captureException(error);
});
```

Обработка ошибок в react-компонентах

Ошибки реакт компонентах в общем случае должны обрабатываться через `React.ErrorBoundary` . Это можно сделать следующим способом:

```
import React from 'react';
import * as Sentry from '@sentry/react';

class MyErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  static getDerivedStateFromError() {
    return { hasError: true };
  }

  componentDidCatch(error, errorInfo) {
    // Custom logic: Send the error to Sentry
    Sentry.captureException(error, { extra: errorInfo });
  }

  render() {
    if (this.state.hasError) {
      // Render your custom fallback UI
      return <h1>Something went wrong.</h1>;
    }

    return this.props.children;
  }
}

export default MyErrorBoundary;
```

Или так же можно обернуть компонент предоставляемый Sentry `ErrorBoudanry` компонент:

```
import React from 'react';
import * as Sentry from '@sentry/react';

const MyComponent1 = () => {};
const WithErrorBoundary = Sentry.withErrorBoundary(MyComponent1);



class MyErrorBoundary extends React.Component {
  ...
}

const MyWrappedBoundary = Sentry.withErrorBoundary(MyComponent, {
  FallbackComponent: MyErrorBoundary,
});

export { WithErrorBoundary, MyWrappedBoundary };
```

Отправка sourcemaps

Чтобы увидеть полноценный stacktrace ошибки нам нужно загрузить в сентри сорсмапы приложения.

Это можно сделать:

1. на этапе сборки с помощью webpack плагина https://docs.sentry.io/platforms/javascript/guides/react/sourcemaps/uploading/webpack/
2. на этапе деплоя: https://docs.sentry.io/platforms/javascript/guides/react/sourcemaps/uploading/cli/

Как видно из документации в обоих случаях нам потребуется подпихивать `SENTRY_AUTH_TOKEN`

Кажется наиболее подходящим вариантом будет инжектить sentry debug id во время сборки с помощью вебпак плагина либо CLI (https://github.com/getsentry/sentry-javascript-bundler-plugins/issues/387),

но для аплоада сорсмапов нам подойдёт скорее второй вариант - во время деплоя, так как на этапе сборки у нас нет доступа в сеть (хотя возможно такой доступ можно прорубить для sentry)

Для этого понадобится ansible script, который будет скачивать образ с сорсмапами и и загржать их на сервер Sentry.

Чтобы вебпак плагин не пытался загружать сорсымапы во время сборки, нужно оставить свойство `sourcemaps` в настройке плагина пустым согласно https://npm.io/package/@sentry/webpack-plugin(но это надо проверять), либо заигнорить все файлы . Если что придётся воспользоваться CLI в дженкинсе для инжекта.