In [1]:

```
import numpy as np
import scipy as sp
from sklearn.decomposition import PCA

import seaborn as sns
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline

sns.set(font_scale=1.3)

red = '#FF3300'
blue = '#0099CC'
green = '#00CC66'
```

Метод главных компонент

(Principal component analysis, PCA)

It had to be U - the SVD song (https://www.youtube.com/watch?v=JEYLfIVvR9I) --- веселая песенка :)

Explained Visually (a setosa project):

Principal Component Analysis (http://setosa.io/ev/principal-component-analysis/)

Eigenvectors and Eigenvalues (http://setosa.io/ev/eigenvectors-and-eigenvalues/)

SVD-разложение

Генерируем датасет

In [2]:

```
X = sp.stats.uniform.rvs(size=(100, 10))
print(X)
```

```
[[2.24084070e-02 7.00103319e-01 2.63010135e-01 2.03580359e-01
  5.49954852e-01 7.02545097e-01 3.68183381e-01 2.60767053e-01
  4.95555721e-01 2.98442510e-01]
 [8.80017678e-01 7.34578065e-01 7.48594445e-01 3.88825026e-01
  7.37351383e-01 4.21592240e-01 3.10308328e-01 8.05600492e-01
  2.46826164e-01 5.65853033e-01]
 [4.57373636e-01 9.52827803e-01 1.28006858e-01 5.29167968e-01
  7.81841934e-01 4.54895379e-01 1.75524858e-01 7.23797660e-01
  1.52313322e-01 2.38744438e-01]
 [8.79424355e-01 8.92249798e-01 7.41546486e-01 4.68253173e-01
  9.87763106e-01 3.71276482e-01 5.67146955e-02 4.01979515e-01
  5.93226129e-01 3.07978983e-01]
 [6.69162116e-01 2.01043136e-01 1.40365214e-02 8.96574696e-01
  8.21204071e-01 1.43875152e-01 4.47468243e-01 4.74077162e-01
  8.29257112e-01 7.61010525e-01]
 [5.69292042e-01 4.15699417e-01 2.34955400e-02 9.16514345e-01
  6.28465854e-01 6.65166514e-02 5.22224813e-01 7.34252138e-02
  6.70150425e-01 8.81686960e-02]
 [1.41605564e-01 3.25534715e-01 8.17183697e-01 7.15028178e-01
```

Вот так можно вычислять сингулярное разложение

In [3]:

```
U, D, V = sp.linalg.svd(X, full_matrices=False)
print(U.shape, D.shape, V.shape)
```

```
(100, 10) (10,) (10, 10)
```

PCA

`sklearn.decomposition.PCA` (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA) `(n_components=None)`

Методы:

- `fit(X)` --- обучиться на данных `X`;
- `fit_transforn(X)` --- обучиться на данных `X` и вернуть сжатое представление `X`;
- `transform(X_new)` --- вернуть сжатое представление `X_new` для обученной ранее модели;
- `inverse_transform(Y)` --- восстановить сжатые данные `Y` в исходное пространство.

Атрибуты:

- `components_` --- главные компоненты в порядке убывания собственных чисел, размер (n_components, n_features);
- `explained_variance_` --- дисперсия вдоль главных компонент, равны собственным числам, размер (n_components,);
- `explained_variance_ratio_` --- доля дисперсии, объясняемая каждой компонентой, размер (n_components,);
- `mean_` --- среднее по данным, размер (n_components,);
- `noise_variance_` --- оценка дисперсии шума для метода Probabilistic PCA.

Другие модификации, реализованные в sklearn:

- `KernelPCA` (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA);
- `SparsePCA` (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.SparsePCA);
- `IncrementalPCA` (https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA).

Генерируем двумерный датасет

In [4]:

```
X = sp.stats.multivariate_normal.rvs(size=150, mean=[0, 3],
                                     cov=[[3, 1], [1, 1]])
```

Применяем PCA с одной главной компонентой

In [5]:

```
pca = PCA(n_components=1)
Y = pca.fit_transform(X)
X_hat = pca.inverse_transform(Y)
```

Его главные компоненты (точнее, одна компонента) -- двумерные векторы

In [6]:

```
pca.components_
```

Out[6]:

```
array([[-0.91866203, -0.3950444 ]])
```

Вектор средних

In [7]:

```
pca.mean_
```

Out[7]:

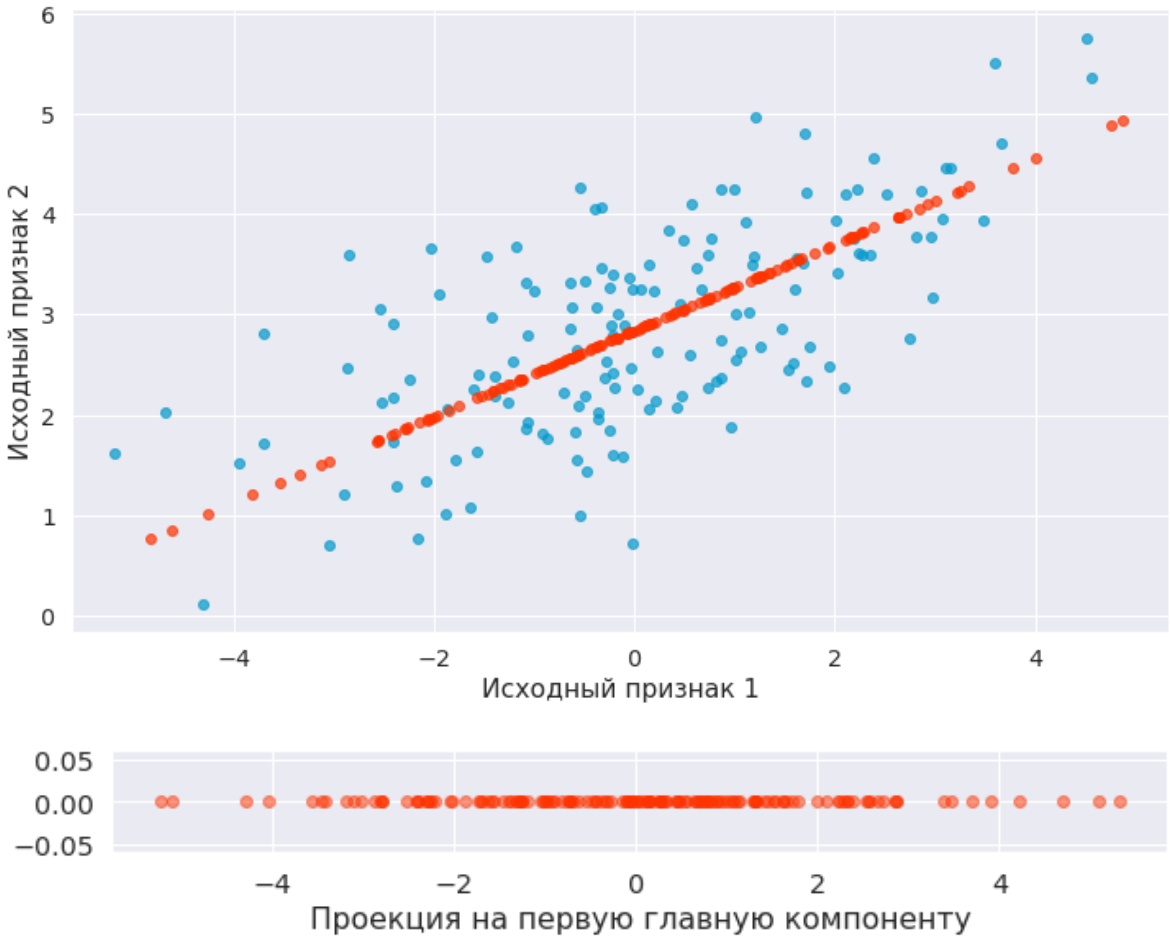
```
array([0.0587315 , 2.86784748])
```

На первом графике синим отмечены исходные точки, красным - они же после проецирования и обратного преобразования. На втором графике точки в одномерном пространстве.

In [8]:

```
plt.figure(figsize=(12, 7))
plt.scatter(X[:, 0], X[:, 1], alpha=0.7, color=blue)
plt.scatter(X_hat[:, 0], X_hat[:, 1], color=red, alpha=0.7)
plt.xlabel('Исходный признак 1')
plt.ylabel('Исходный признак 2')
plt.axis('equal')
plt.show()

plt.figure(figsize=(10, 1))
plt.scatter(Y, np.zeros(len(Y)), alpha=0.5, color=red)
plt.xlabel('Проекция на первую главную компоненту')
plt.show()
```



Пусть есть два хорошо разделимых класса

In [9]:

```
a = sp.stats.multivariate_normal.rvs(size=150, mean=[4, 3],
                                     cov=[[2, 1], [1, 2]])
b = sp.stats.multivariate_normal.rvs(size=150, mean=[0, 6],
                                     cov=[[2, 1], [1, 2]])
X = np.vstack([a, b])
c = np.hstack([np.zeros(len(a)), np.ones(len(b))])

plt.figure(figsize=(12, 9))
plt.scatter(X[:, 0], X[:, 1], c=c, alpha=0.7, cmap='rainbow')
plt.xlabel('Исходный признак 1')
plt.ylabel('Исходный признак 2')
plt.xlim((-5, 9)), plt.ylim((-1, 10))
plt.axis('equal')
plt.show()
```

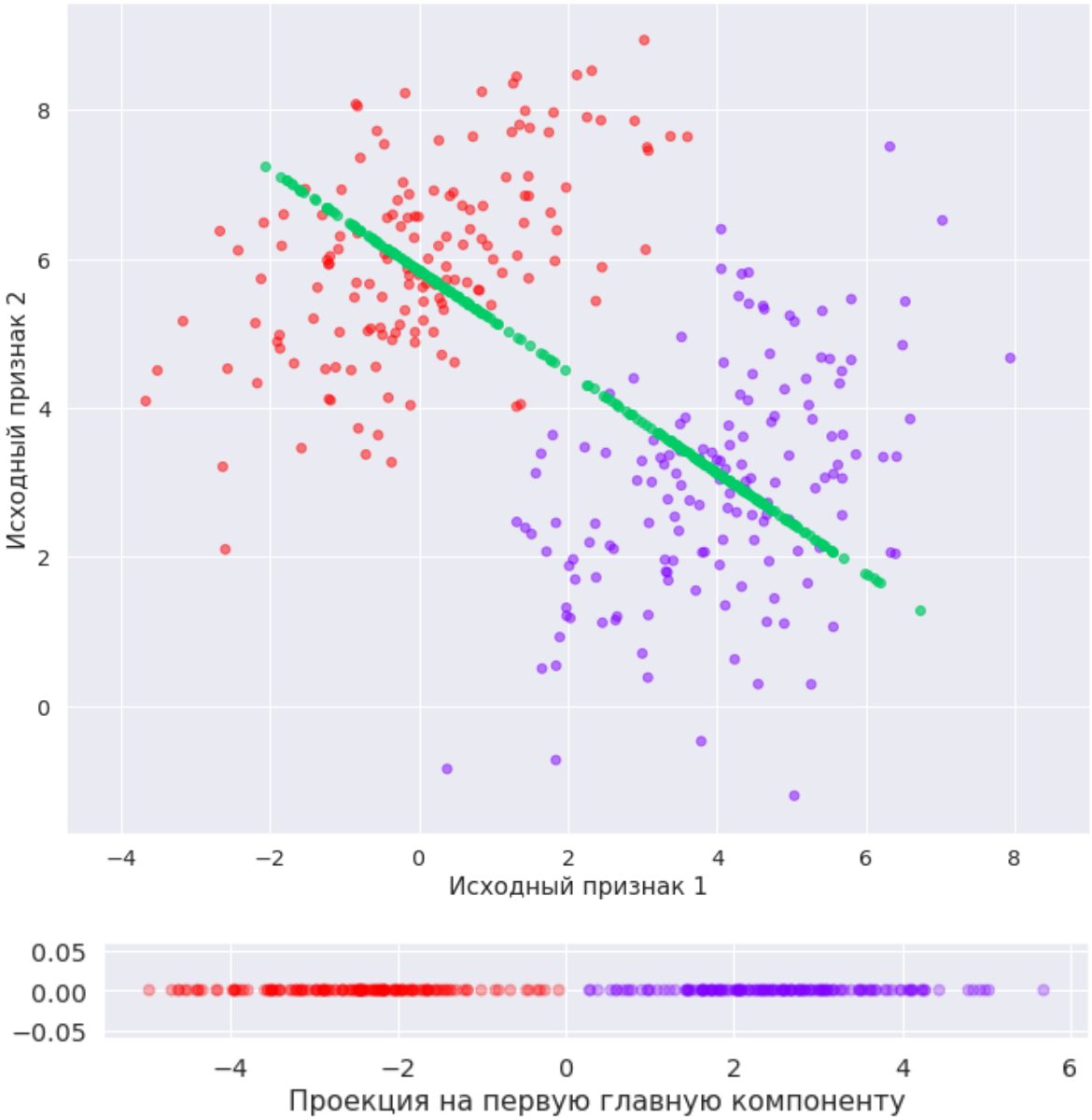
Проецируем на одномерное подпространство, тут хорошо заметна пространственная структура классов.

In [10]:

```
pca = PCA(n_components=1)
Y = pca.fit_transform(X)
X_hat = pca.inverse_transform(Y)

plt.figure(figsize=(12, 10))
plt.scatter(X[:, 0], X[:, 1], c=c, alpha=0.5, cmap='rainbow')
plt.scatter(X_hat[:, 0], X_hat[:, 1], c=green, alpha=0.7)
plt.xlabel('Исходный признак 1')
plt.ylabel('Исходный признак 2')
plt.xlim((-5, 9)), plt.ylim((-1, 10))
plt.axis('equal')
plt.show()

plt.figure(figsize=(10, 1))
plt.scatter(Y, np.zeros(len(Y)), c=c, alpha=0.3, cmap='rainbow')
plt.xlabel('Проекция на первую главную компоненту')
plt.show()
```

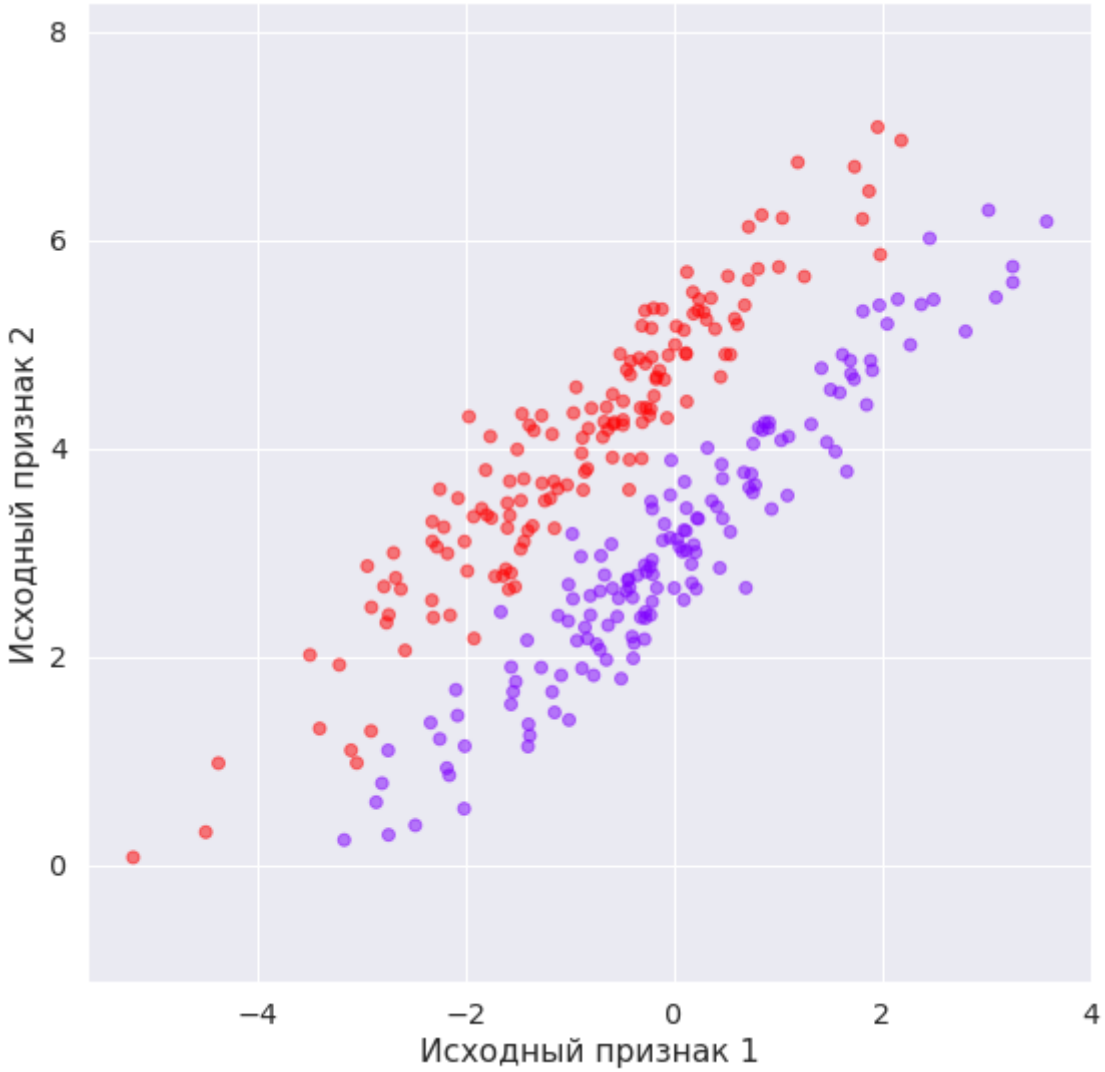


А что, если два вытянутых класса, расположенных близко, как на рисунке ниже?

In [11]:

```
a = sp.stats.multivariate_normal.rvs(size=150, mean=[0, 3],
                                     cov=[[2, 1.9], [1.9, 2]])
b = sp.stats.multivariate_normal.rvs(size=150, mean=[-1, 4],
                                     cov=[[2, 1.9], [1.9, 2]])
X = np.vstack([a, b])
c = np.hstack([np.zeros(len(a)), np.ones(len(b))])

plt.figure(figsize=(9, 9))
plt.scatter(X[:, 0], X[:, 1], c=c, alpha=0.5, cmap='rainbow')
plt.xlabel('Исходный признак 1')
plt.ylabel('Исходный признак 2')
plt.xlim((-5, 5)), plt.ylim((-1, 9))
plt.axis('equal')
plt.show()
```



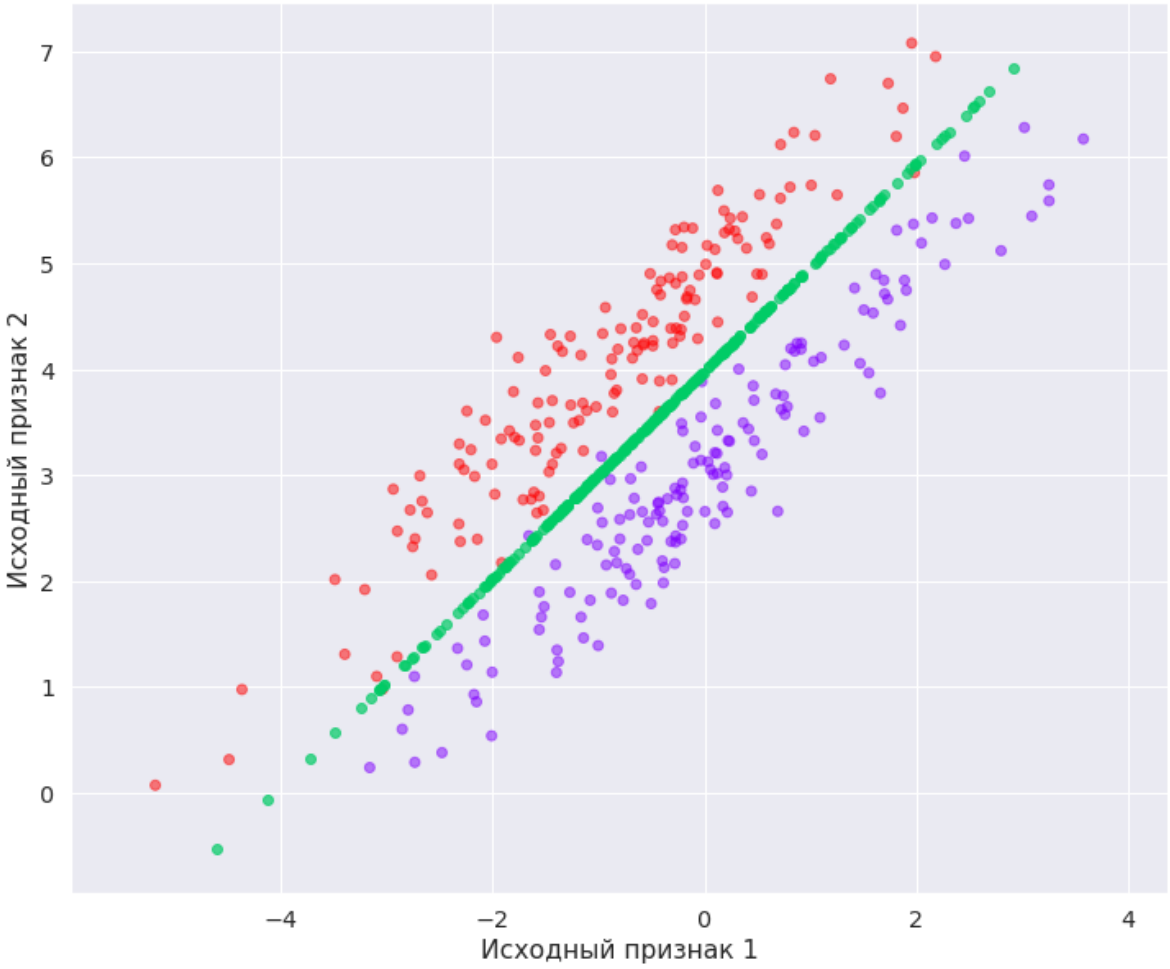
Главная компонента направлена вдоль этих классов, и при проецировании точки классов смешаются.

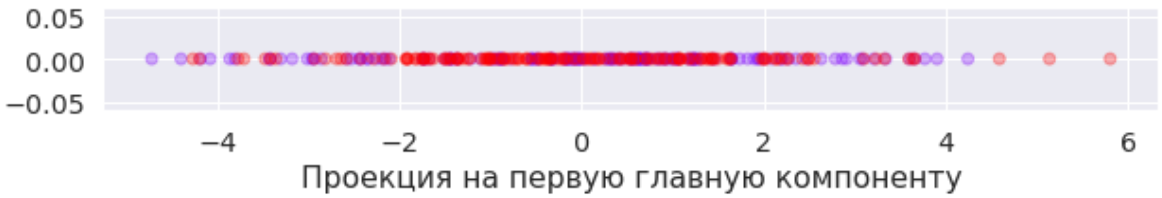
In [12]:

```
pca = PCA(n_components=1)
Y = pca.fit_transform(X)
X_hat = pca.inverse_transform(Y)

plt.figure(figsize=(12, 10))
plt.scatter(X[:, 0], X[:, 1], c=c, alpha=0.5, cmap='rainbow')
plt.scatter(X_hat[:, 0], X_hat[:, 1], c=green, alpha=0.7)
plt.xlabel('Исходный признак 1')
plt.ylabel('Исходный признак 2')
plt.xlim((-5, 5)), plt.ylim((-1, 9))
plt.axis('equal')
plt.show()

plt.figure(figsize=(10, 1))
plt.scatter(Y, np.zeros(len(Y)), c=c, alpha=0.3, cmap='rainbow')
plt.xlabel('Проекция на первую главную компоненту')
plt.show()
```





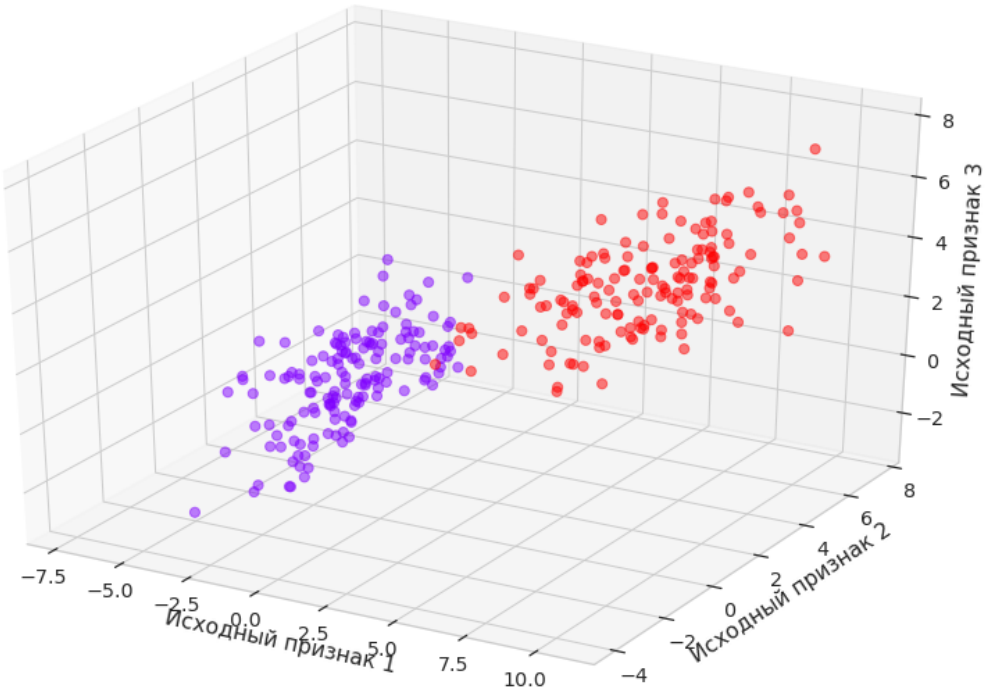
In [13]:

```
a = sp.stats.multivariate_normal.rvs(size=150, mean=[-2, 1, 0],
                                     cov=[[3, -1, 1],
                                          [-1, 4, 0.5],
                                          [1, 0.5, 1.5]])
b = sp.stats.multivariate_normal.rvs(size=150, mean=[7, 3, 4],
                                     cov=[[3, 2, 1],
                                          [2, 4, 0.5],
                                          [1, 0.5, 1.5]])
X = np.vstack([a, b])
c = np.hstack([np.zeros(len(a)), np.ones(len(b))])
```

In [14]:

```
sns.set(font_scale=1.3, style='whitegrid')

fig = plt.figure(figsize=(15, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2],
           s=50, c=c, alpha=0.5, cmap='rainbow')
ax.set_xlabel('Исходный признак 1')
ax.set_ylabel('Исходный признак 2')
ax.set_zlabel('Исходный признак 3')
# ax.axis('equal')
plt.show()
```

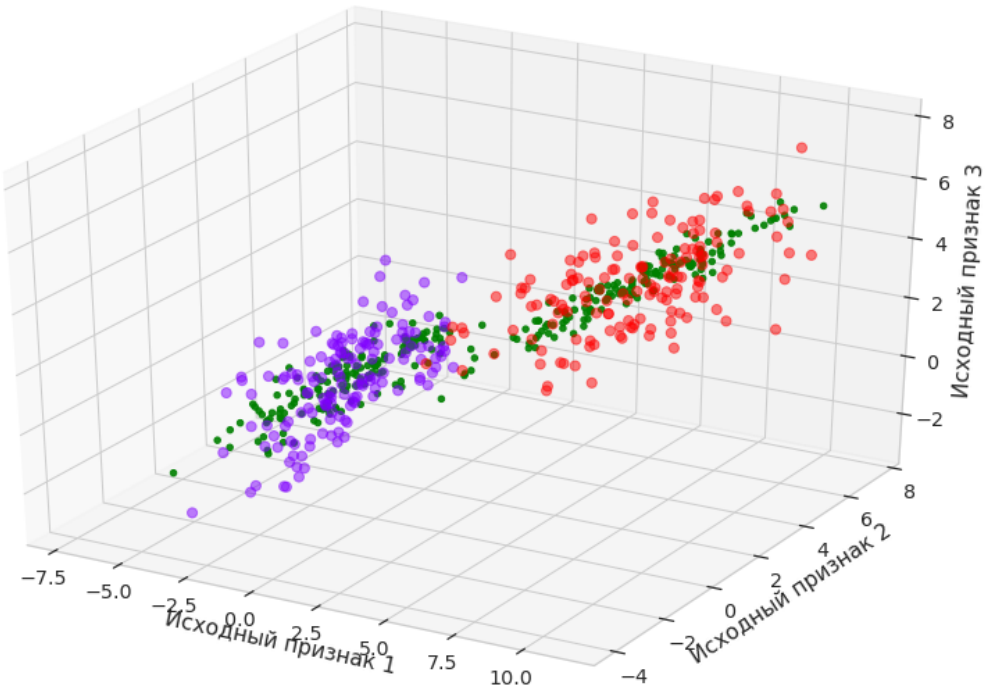


In [15]:

```
pca = PCA(n_components=2)
Y = pca.fit_transform(X)
X_hat = pca.inverse_transform(Y)

fig = plt.figure(figsize=(15, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2],
           s=50, c=c, alpha=0.5, cmap='rainbow')
ax.scatter(X_hat[:, 0], X_hat[:, 1], X_hat[:, 2],
           s=20, c='green', alpha=0.9)
ax.set_xlabel('Исходный признак 1')
ax.set_ylabel('Исходный признак 2')
ax.set_zlabel('Исходный признак 3')
# ax.axis('equal')
plt.show()

sns.set(font_scale=1.3)
plt.figure(figsize=(6, 6))
plt.scatter(Y[:, 0], Y[:, 1], c=c, alpha=0.3, cmap='rainbow')
plt.xlabel('Проекция на первую главную компоненту')
plt.ylabel('Проекция на вторую главную компоненту')
plt.axis('equal')
plt.show()
```



In [16]:

```
1 import plotly
2 import plotly.graph_objs as go
3 plotly.offline.init_notebook_mode()
```

In [17]:

```
fig = go.Figure()
fig.update_layout(
        autosize=False,
        width=1000,
        height=1000,
)

fig.add_trace(go.Scatter3d(
    x=X[c==0, 0], y=X[c==0, 1], z=X[c==0, 2], mode='markers',
    marker={'size': 7, 'opacity': 0.65, 'color': red}
))

fig.add_trace(go.Scatter3d(
    x=X[c==1, 0], y=X[c==1, 1], z=X[c==1, 2], mode='markers',
    marker={'size': 7, 'opacity': 0.65, 'color': blue}
))

fig.add_trace(go.Scatter3d(
    x=X_hat[:, 0], y=X_hat[:, 1], z=X_hat[:, 2], mode='markers',
    marker={'size': 4, 'opacity': 0.9, 'color': green}
))

fig.update_layout(
        margin={'l': 0, 'r': 0, 'b': 0, 't': 0},
        scene = dict(
            xaxis_title='Признак 1',
            yaxis_title='Признак 2',
            zaxis_title='Признак 3',
        )
    )

# Отображение.
plotly.offline.iplot(fig)
```

Другие методы снижения размерности

http://scikit-learn.org/stable/modules/manifold.html#manifold (http://scikit-learn.org/stable/modules/manifold.html#manifold)

Примеры:

http://scikit-learn.org/stable/auto_examples/manifold/plot_compare_methods.html#sphx-glr-auto-examples-manifold-plot-compare-methods-py (http://scikit-learn.org/stable/auto_examples/manifold/plot_compare_methods.html#sphx-glr-auto-examples-manifold-plot-compare-methods-py)

http://scikit-learn.org/stable/auto_examples/manifold/plot_manifold_sphere.html#sphx-glr-auto-examples-manifold-plot-manifold-sphere-py (http://scikit-learn.org/stable/auto_examples/manifold/plot_manifold_sphere.html#sphx-glr-auto-examples-manifold-plot-manifold-sphere-py)