Модель классификации статуса стартапа.

Описание проекта:

Проект представляет собой модель классификации стартапов, предсказывающую вероятность их закрытия. Данные для проекта взяты с платформы Crunchbase.

Основные этапы исследования:

- загрузка и ознакомление с данными
- предварительная обработка
- разведочный анализ
- разработка новых синтетических признаков
- проверка на мультиколлинеарность
- отбор финального набора обучающих признаков
- выбор и обучение моделей
- итоговая оценка качества предсказания лучшей модели
- анализ важности ее признаков
- отчет по исследованию

In [1]:
```
#загрузка библиотек для работы с данными
import numpy as np
import pandas as pd

#загрузка библиотек для анализа корреляции
import phik

#загрузка библиотек для работы с графиками
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# загружаем класс pipeline
from sklearn.pipeline import Pipeline


# загружаем классы для подготовки данных
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import (OneHotEncoder,
                                   OrdinalEncoder,
                                   StandardScaler,
                                   RobustScaler,
                                   MinMaxScaler)

from sklearn.compose import ColumnTransformer

# загружаем класс для работы с пропусками
from sklearn.impute import SimpleImputer

# загружаем функцию для работы с метриками
from sklearn.metrics import (roc_auc_score,
                             accuracy_score,
                             f1_score)

# импортируем класс GridSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

# загружаем нужные модели
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from catboost import CatBoostClassifier

#загружаем модуль проверки признаков
import shap
```

Загрузка, изучение данных, преподготовка данных

Загрузка данных

In [2]:
```
train_df = pd.read_csv('kaggle_startups_train_01.csv')
test_df = pd.read_csv('kaggle_startups_test_01.csv')
```

Изучение данных

Изучение тренировачных данных

In [3]:
```
#Вывод основной информации о датафрейме
def data_exploration(dataframe):
    print('\033[0m')
    print('\033[1m' + 'Информация о датафрейме:')
    print('\033[0m')
    display(dataframe.info())
    print('\033[0m')
    print('\033[1m' + 'Первые пять строк:')
    print('\033[0m')
    display(dataframe.head())
```

In [4]:
```
data_exploration(train_df)
```

Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52879 entries, 0 to 52878
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   name               52878 non-null  object 
 1   category_list      50374 non-null  object 
 2   funding_total_usd  42753 non-null  float64
 3   status             52879 non-null  object 
 4   country_code       47351 non-null  object 
 5   state_code         46082 non-null  object 
 6   region             46489 non-null  object 
 7   city               46489 non-null  object 
 8   funding_rounds     52879 non-null  int64  
 9   founded_at         52879 non-null  object 
 10  first_funding_at   52858 non-null  object 
 11  last_funding_at    52879 non-null  object 
 12  closed_at          4962 non-null   object 
dtypes: float64(1), int64(1), object(11)
memory usage: 5.2+ MB
None
```

Первые пять строк:

	name	category_list	funding_total_usd	status	country_code	state_code	region
0	Lunchgate	Online Reservations|Restaurants	828626.0	operating	CHE	25	Zurich
1	EarLens	Manufacturing|Medical|Medical Devices	42935019.0	operating	USA	CA	SF Bay Area
2	Reviva Pharmaceuticals	Biotechnology	35456381.0	operating	USA	CA	SF Bay Area
3	Sancilio and Company	Health Care	22250000.0	operating	NaN	NaN	NaN
4	WireTough Cylinders	Manufacturing	NaN	operating	USA	VA	VA - Other

- **name** - Название стартапа

- **category_list** - Индустрия в которой работает стартап
- **funding_total_usd** - Объем инвестиций
- **status** - статус функционирования (целевой признак)
- **country_code** - страна
- **state_code** - штат
- **region** - Регион стартапа
- **city** - Город где функционирует стартап
- **funding_rounds** - число раундов финансирования
- **founded_at** - дата создания стартапа
- **first_funding_at** - дата первого раунда финансирования
- **last_funding_at** - дата последнего раунда финансирования
- **closed_at** - дата закрытия стартапа

Изучение тестовых данных

In [5]:
```
data_exploration(test_df)
```

Информация о датафрейме:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13211 entries, 0 to 13210
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   name               13211 non-null  object 
 1   category_list      12610 non-null  object 
 2   funding_total_usd  10616 non-null  float64
 3   country_code       11827 non-null  object 
 4   state_code         11512 non-null  object 
 5   region             11618 non-null  object 
 6   city               11620 non-null  object 
 7   funding_rounds     13211 non-null  int64  
 8   founded_at         13211 non-null  object 
 9   first_funding_at   13211 non-null  object 
 10  last_funding_at    13211 non-null  object 
 11  closed_at          1234 non-null   object 
dtypes: float64(1), int64(1), object(10)
memory usage: 1.2+ MB
None
```

Первые пять строк:

	name	category_list	funding_total_usd	country_code	state_code	region	city	funding_
0	Crystalsol	Clean Technology	2819200.0	NIC	17	NaN	NaN	
1	JBI Fish & Wings	Hospitality	NaN	USA	TN	TN - Other	Humboldt	
2	COINPLUS	Finance	428257.0	LUX	3	Esch-sur-alzette	Esch-sur-alzette	
3	Imagine Communications	Software|Video|Video Streaming	34700000.0	USA	CA	San Diego	San Diego	
4	DNA13	Software	4530000.0	CAN	ON	Ottawa	Ottawa	

Заключение по ознакомлению с данными

- Тренировочные данные представлены таблицей размерностью 52879 строк и 13 столбцов
- Тестовые данные представлены 13211 строк и 12 столбцов
- Содержание столбцов схоже, в тестовых данных отсутствует столбец с целевым признаком

Предподготовка данных

Изменение типа данных

Изменение типа данных в тренировочном датасете

In [6]:
```
#Поменяем тип данных для дат
train_df['founded_at'] = pd.to_datetime(train_df['founded_at'], format='%Y-%m-%d')
train_df['first_funding_at'] = pd.to_datetime(train_df['first_funding_at'], format='%Y-%m-%d'
train_df['last_funding_at'] = pd.to_datetime(train_df['last_funding_at'], format='%Y-%m-%d')
train_df['closed_at'] = pd.to_datetime(train_df['closed_at'], format='%Y-%m-%d')
```

Изменение типа данных в тестовом датасете

In [7]:
```
#Поменяем тип данных для дат
test_df['founded_at'] = pd.to_datetime(test_df['founded_at'], format='%Y-%m-%d')
test_df['first_funding_at'] = pd.to_datetime(test_df['first_funding_at'], format='%Y-%m-%d')
test_df['last_funding_at'] = pd.to_datetime(test_df['last_funding_at'], format='%Y-%m-%d')
test_df['closed_at'] = pd.to_datetime(test_df['closed_at'], format='%Y-%m-%d')
```

Поиск пропусков и дубликатов

Поиск пропусков и дубликатов в тренировочном файле

In [8]:
```
#Проверка на отсутствие пропусков и дупликатов
print('Пропуски в строках таблицы train_df')
display(pd.DataFrame((round(train_df.isna().mean()*100)))
        .style.background_gradient('coolwarm'))
print('Дубликаты train_df', train_df.duplicated().sum())
```

```
Пропуски в строках таблицы train_df
```

	0
name	0.000000
category_list	5.000000
funding_total_usd	19.000000
status	0.000000
country_code	10.000000
state_code	13.000000
region	12.000000
city	12.000000
funding_rounds	0.000000
founded_at	0.000000
first_funding_at	0.000000
last_funding_at	0.000000
closed_at	91.000000

```
Дубликаты train_df 0
```

- В тренировочном файле дубликаты не выявлены
- 91% наблюдений не содержит информацию о том, когда закрылись компании, а следовательно предполагаем, что они еще работают.
- 5% наблюдений не содержат информацию в каком рынке функционирует стартап
- 10% не содержит информацию о стране происхождения
- 13% о штате происхождения
- 12% не содержин информацию о регионе происхождения
- 12% не содержит информацию о городе происхождения
- Дубликаты не выявлены

Проведем дополнительный анализ об отсутствующей информации для закрытых стартапов

In [9]:
```
#Проверка на пропуски в закрытых стартапах
print('Пропуски в строках таблицы train_df')
display(pd.DataFrame(round(train_df[train_df['status'] == 'closed'].isna().mean()*100,))
        .style.background_gradient('coolwarm'))
```

```
Пропуски в строках таблицы train_df
```

	0
name	0.000000
category_list	15.000000
funding_total_usd	21.000000
status	0.000000
country_code	26.000000
state_code	28.000000
region	28.000000
city	28.000000
funding_rounds	0.000000
founded_at	0.000000
first_funding_at	0.000000
last_funding_at	0.000000
closed_at	0.000000

Похоже, что для закрытых стартапов характерно отсутствие информации об их происхождении. Возможно использовать данный инсайт для создания нового признака.

In [10]:
```
#Заменим отсутствующую категорийную информацию на заглушку для категории
train_df['category_list'] = train_df['category_list'].fillna('Unknown_category')
train_df['country_code'] = train_df['country_code'].fillna('Unknown_country')
train_df['state_code'] = train_df['state_code'].fillna('Unknown_state')
train_df['region'] = train_df['region'].fillna('Unknown_region')
train_df['city'] = train_df['city'].fillna('Unknown_city')

#Заменим отсутствующую информацию о финансировании на 0
train_df['funding_total_usd'] = train_df['funding_total_usd'].fillna(0)

#Заменим отсутствующую временную информацию на заглушку для категории
train_df.closed_at.fillna(pd.to_datetime('2018-01-01'), inplace=True)
```

In [11]:
```
#Проверка на отсутствие пропусков и дупликатов
print('Пропуски в строках таблицы train_df')
display(pd.DataFrame(round(train_df.isna().mean()*100,))
        .style.background_gradient('coolwarm'))
print('Дубликаты train_df', train_df.duplicated().sum())
```

```
Пропуски в строках таблицы train_df
```

	0
name	0.000000
category_list	0.000000
funding_total_usd	0.000000
status	0.000000
country_code	0.000000
state_code	0.000000
region	0.000000
city	0.000000
funding_rounds	0.000000
founded_at	0.000000
first_funding_at	0.000000
last_funding_at	0.000000
closed_at	0.000000

```
Дубликаты train_df 0
```

Поиск пропусков и дубликатов в тестовом файле

In [12]:
```
#Проверка на отсутствие пропусков и дупликатов
print('Пропуски в строках таблицы test_df')
display(pd.DataFrame(round(test_df.isna().mean()*100,))
        .style.background_gradient('coolwarm'))
print('Дубликаты test_df', test_df.duplicated().sum())
```

```
Пропуски в строках таблицы test_df
```

	0
name	0.000000
category_list	5.000000
funding_total_usd	20.000000
country_code	10.000000
state_code	13.000000
region	12.000000
city	12.000000
funding_rounds	0.000000
founded_at	0.000000
first_funding_at	0.000000
last_funding_at	0.000000
closed_at	91.000000

```
Дубликаты test_df 0
```

- В тренировочном файле дубликаты не выявлены
- 91% наблюдений не содержит информацию о том, когда закрылись компании, а следовательно предполагаем, что они еще работают.
- 5% наблюдений не содержат информацию в каком рынке функционирует стартап
- 10% не содержит информацию о стране происхождения
- 12% о штате происхождения
- 12% не содержин информацию о регионе происхождения
- 12% не содержит информацию о городе происхождения
- Дубликаты не выявлены

Датасет имеет аналогичный объем пропусков

In [13]:
```
#Заменим отсутствующую категорийную информацию на заглушку для категории
test_df['category_list'] = test_df['category_list'].fillna('Unknown_category')
test_df['country_code'] = test_df['country_code'].fillna('Unknown_country')
test_df['state_code'] = test_df['state_code'].fillna('Unknown_state')
test_df['region'] = test_df['region'].fillna('Unknown_region')
test_df['city'] = test_df['city'].fillna('Unknown_city')

#Заменим отсутствующую информацию о финансировании на 0
test_df['funding_total_usd'] = test_df['funding_total_usd'].fillna(0)

#Заменим отсутствующие даты в столбце closed_at на дату загрузки данных (2018-01-01)
test_df['closed_at'] = test_df['closed_at'].fillna('2018-01-01')
```

In [14]:
```
#Проверка на отсутствие пропусков и дупликатов
print('Пропуски в строках таблицы test_df')
display(pd.DataFrame(round(test_df.isna().mean()*100,))
        .style.background_gradient('coolwarm'))
print('Дубликаты test_df', test_df.duplicated().sum())
```

```
Пропуски в строках таблицы test_df
```

	0
name	0.000000
category_list	0.000000
funding_total_usd	0.000000
country_code	0.000000
state_code	0.000000
region	0.000000
city	0.000000
funding_rounds	0.000000
founded_at	0.000000
first_funding_at	0.000000
last_funding_at	0.000000
closed_at	0.000000

```
Дубликаты test_df 0
```

Заключение по предподготовке данных

Тренировочный датасет и тестовый датасет имеют схожую статистику по пропускам

- В тренировочном файле дубликаты не выявлены
- 91% наблюдений не содержит информацию о том, когда закрылись компании, а следовательно предполагаем, что они еще работают.
- 5% наблюдений не содержат информацию в каком рынке функционирует стартап
- 10% не содержит информацию о стране происхождения
- 13% о штате происхождения
- 12% не содержин информацию о регионе происхождения
- 12% не содержит информацию о городе происхождения

- Отсутствующие данные заменены на заглушки

Создание новых признаков

Создадим новые признаки на базе столбцов с датами:

- lifetime продолжительность жизни стартапа от даты создания до даты закрытия
- funding_length продолжительность жизни стартапа между первым и последним раундом финансирования
- first_funding_lt продолжительность жизни стратапа между первым финансированием и созданием

- funding_life_share - доля жизни стартапа с финансированием

In [15]:
```
# Сосчитаем разницу между датами
train_df['lifetime'] = (train_df['closed_at'] - train_df['founded_at']).dt.days
train_df['funding_length'] = (train_df['last_funding_at'] - train_df['first_funding_at']).dt.
train_df['first_funding_lt'] = (train_df['first_funding_at']-train_df['founded_at']).dt.days
train_df['funding_life_share'] = train_df['funding_length'] / train_df['lifetime']
```

In [16]:
```
train_df.head(10)
```

Out[16]:

	name	category_list	funding_total_usd	status	country_code	state_code	
0	Lunchgate	Online Reservations|Restaurants	828626.0	operating	CHE	25	
1	EarLens	Manufacturing|Medical|Medical Devices	42935019.0	operating	USA	CA	
2	Reviva Pharmaceuticals	Biotechnology	35456381.0	operating	USA	CA	
3	Sancilio and Company	Health Care	22250000.0	operating	Unknown_country	Unknown_state	U
4	WireTough Cylinders	Manufacturing	0.0	operating	USA	VA	
5	Connected Sports Ventures	Mobile	4300000.0	operating	USA	NJ	
6	Attensity	Analytics|Business Analytics|Social CRM|Social...	90000000.0	operating	USA	CA	
7	Mesh Networks	Software	4300000.0	operating	USA	TX	
8	AngioScore	Biotechnology	42000000.0	operating	USA	CA	
9	Vidatronic	Semiconductors	1250500.0	operating	USA	TX	

In [17]:
```
# Сосчитаем разницу между датами для тестовых данных
test_df['lifetime'] = (test_df['closed_at'] - test_df['founded_at']).dt.days
test_df['funding_length'] = (test_df['last_funding_at'] - test_df['first_funding_at']).dt.day
test_df['funding_life_share'] = test_df['funding_length'] / test_df['lifetime']
test_df['first_funding_lt'] = (test_df['first_funding_at']-test_df['founded_at']).dt.days
```

Исследовательский анализ данных

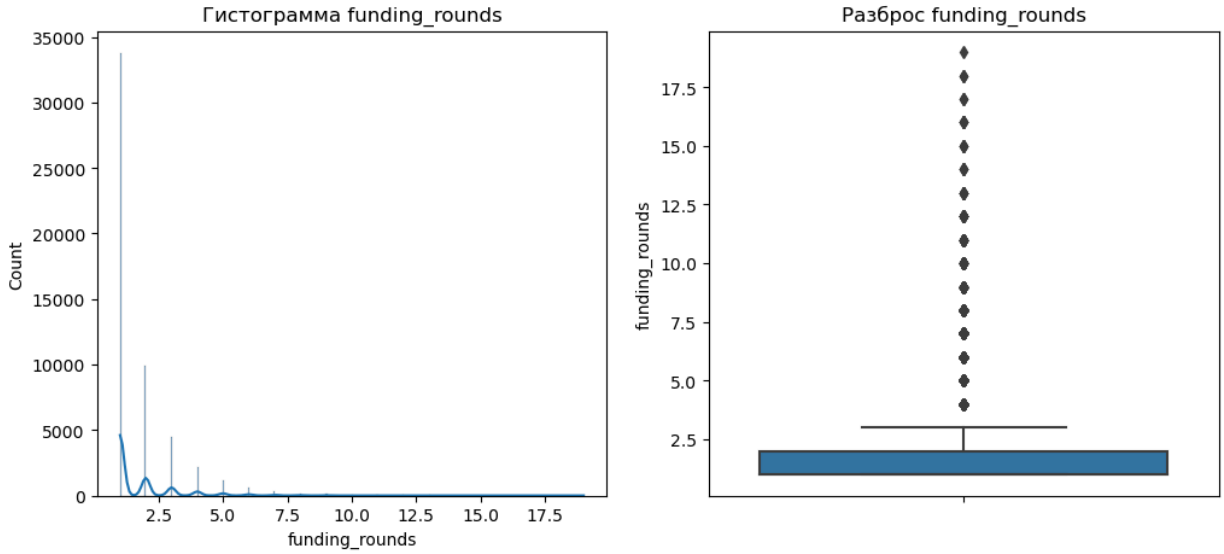
Изучение количественных данных

In [18]:
```
# Функция для вычисления нижнего и верхнего пределов для определения выбросов
def outlier_detection(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_range = Q1 - (1.5 * IQR)
    upper_range = Q3 + (1.5 * IQR)
    return lower_range, upper_range


# Функция для вывода анализа количественных данных
def plot_numerical_features(dataframe, features):
    for feature in features:
        if feature != 'id' and pd.api.types.is_numeric_dtype(dataframe[feature]):
            #Построение гистограммы и ящика с усами
            fig, axes = plt.subplots(1, 2, figsize=(12, 5))
            sns.histplot(dataframe[feature], kde=True, ax=axes[0], log_scale=False, palette="
            axes[0].set_title(f'Гистограмма {feature}')
            sns.boxplot(y=dataframe[feature], ax=axes[1])
            axes[1].set_title(f'Разброс {feature}')
            plt.show()
            # Отображение описательной статистики
```

```
        display(dataframe[feature].describe().to_frame().T)

        # Вычисление и вывод нижнего и верхнего пределов для выбросов
        lower_range, upper_range = outlier_detection(dataframe[feature])
        print(f'Нижняя граница выбросов признака {feature}: {lower_range}')
        print(f'Верхняя граница выбросов признака {feature}: {upper_range}')

        # Подсчет значений за пределами границ
        below_lower = dataframe[dataframe[feature] < lower_range].shape[0]
        above_upper = dataframe[dataframe[feature] > upper_range].shape[0]
        print(f'Количество значений за нижней границей {feature}: {below_lower}')
```

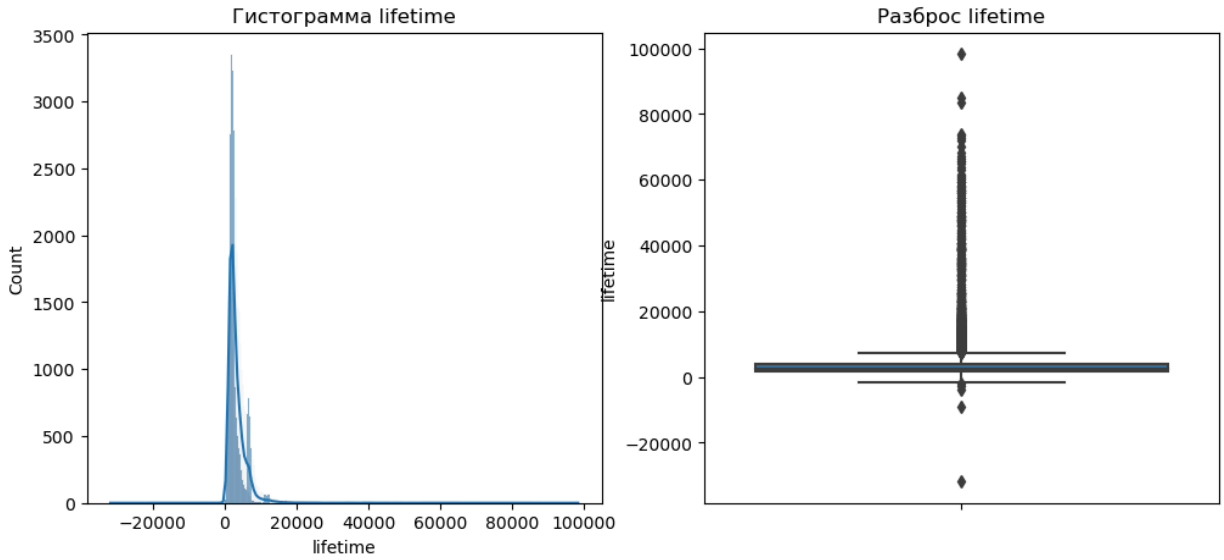
In [19]:
```
numerical_columns_dea = train_df.select_dtypes(include=['number'], exclude=['object', 'dateti
numerical_columns_dea.remove('funding_total_usd')
```

In [20]:
```
plot_numerical_features(train_df, numerical_columns_dea)
```



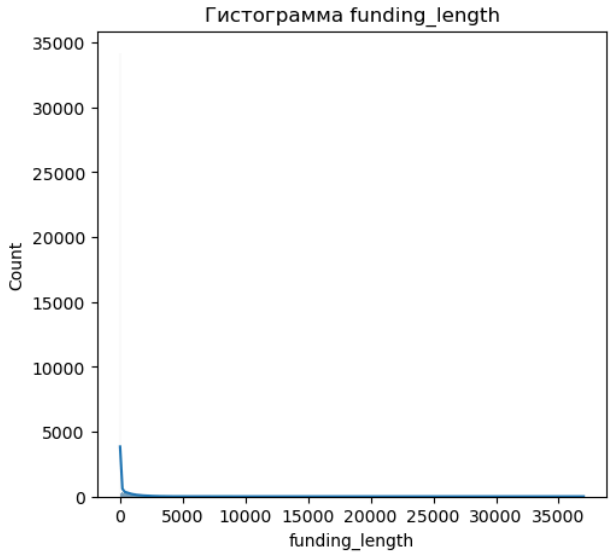
	count	mean	std	min	25%	50%	75%	max
funding_rounds	52879.0	1.73827	1.371993	1.0	1.0	1.0	2.0	19.0

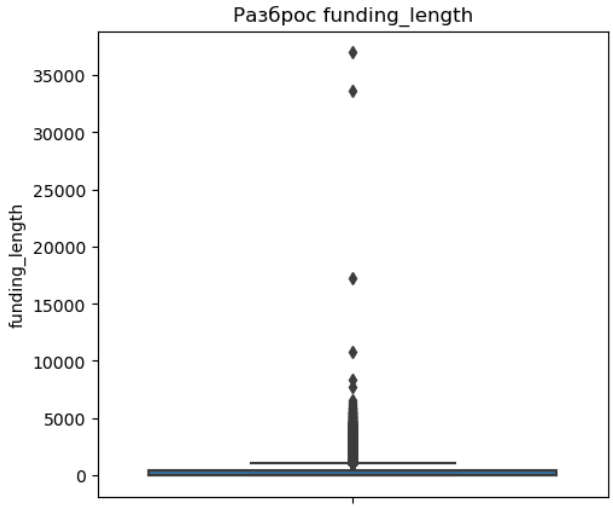
```
Нижняя граница выбросов признака funding_rounds: -0.5
Верхняя граница выбросов признака funding_rounds: 3.5
Количество значений за нижней границей funding_rounds: 0
```



	count	mean	std	min	25%	50%	75%	max
lifetime	52879.0	3317.351406	3396.031285	-31823.0	1740.0	2557.0	4018.0	98250.0

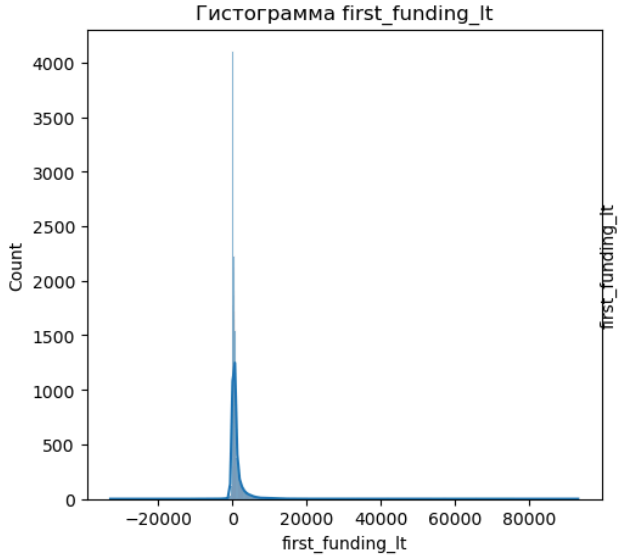
```
Нижняя граница выбросов признака lifetime: -1677.0
Верхняя граница выбросов признака lifetime: 7435.0
Количество значений за нижней границей lifetime: 5
```

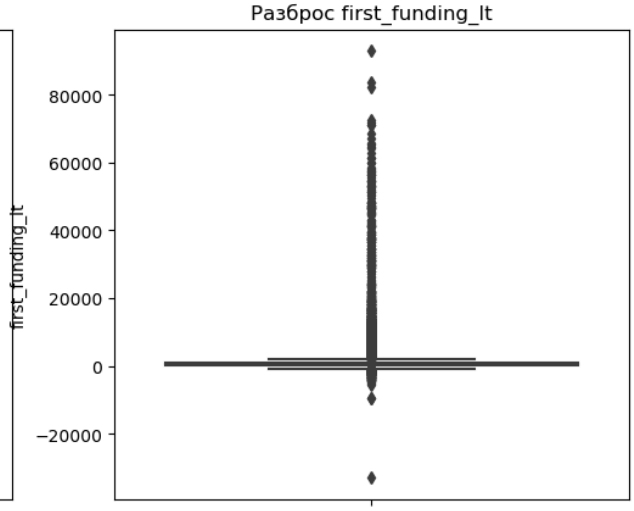




	count	mean	std	min	25%	50%	75%	max
funding_length	52858.0	346.647092	724.049585	0.0	0.0	0.0	425.0	36994.0

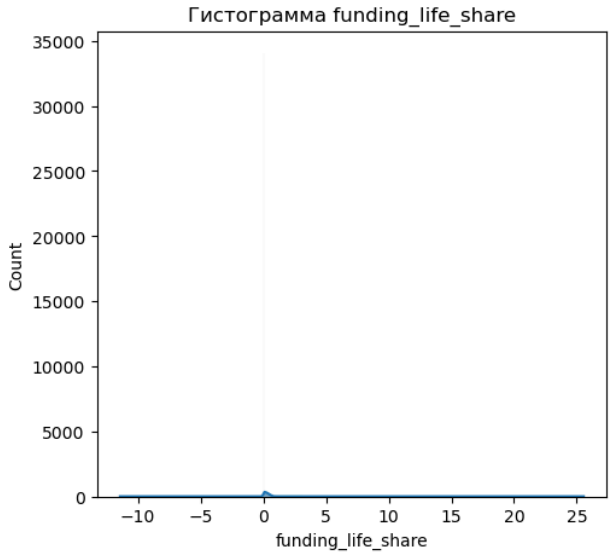
```
Нижняя граница выбросов признака funding_length: -637.5
Верхняя граница выбросов признака funding_length: 1062.5
Количество значений за нижней границей funding_length: 0
```

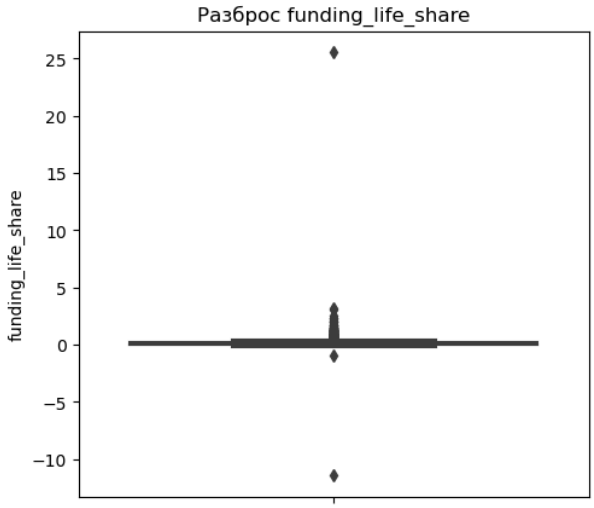




	count	mean	std	min	25%	50%	75%	max
first_funding_lt	52858.0	1162.303965	3113.476082	-32919.0	216.0	482.0	1003.0	93053.0

```
Нижняя граница выбросов признака first_funding_lt: -964.5
Верхняя граница выбросов признака first_funding_lt: 2183.5
Количество значений за нижней границей first_funding_lt: 209
```





	count	mean	std	min	25%	50%	75%	max
funding_life_share	52858.0	0.102334	0.21782	-11.421053	0.0	0.0	0.158321	25.548343

```
Нижняя граница выбросов признака funding_life_share: -0.2374818080952175
Верхняя граница выбросов признака funding_life_share: 0.3958030134920292
Количество значений за нижней границей funding_life_share: 2
```

Выводы:

- В среднем 1 раунд финансирования у всех наблюдений
- Присутствует отрицательный lifetime, скорее всего это аномалия
- В среднем между первым и последним финансированием проходит 346 дней
- Присутствуют отрицательные значения времени до первого фандинга перед созданием компании, возможно финансирование поступило на pre seed фазе

In [21]:
```
#Найдем количество строк
print('Строк с отрицательной продолжительностью', (train_df['lifetime'] < 0).sum())
```

```
Строк с отрицательной продолжительностью 28
```

In [22]:
```
#Убираем строки
train_df = train_df[train_df['lifetime'] >= 0]
```

In [23]:
```
#Сделаем новый признак указывающий на каком моменте получена первая инвестиция (до создания ю
train_df['funding_stage'] = np.where(train_df['first_funding_lt'] < 0, 'pre-seed', 'after see
test_df['funding_stage'] = np.where(test_df['first_funding_lt'] < 0, 'pre-seed', 'after seed'
```

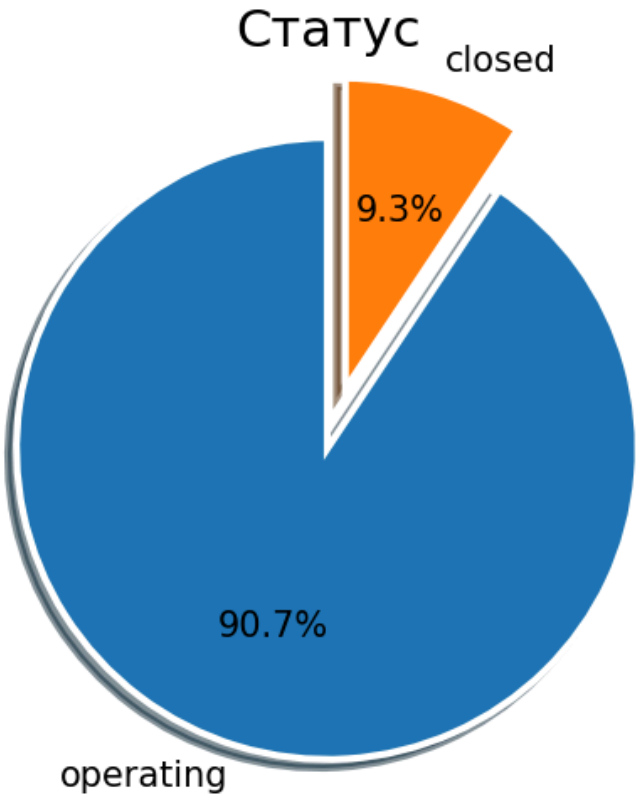
Изучение качественных данных

In [24]:
```
#Функция для анализа категориальных данных
def plot_column_data(column_name):
    # Определение статуса целевого признака, None - формула будет принята для всего массива
    statuses = [None, 'operating', 'closed']

    fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(24, 8), constrained_layout=True)

    for i, status in enumerate(statuses):
        # Фильтрация при уточненном статусе
        if status:
            filtered_df = train_df[train_df['status'] == status.lower()]
        else:
            filtered_df = train_df  # На весь датафрейм если не интересует статус

        # Шаг 1: Разделение и расширение указанного столбца на индивидуальные категории, если
        if filtered_df[column_name].dtype == object and '|' in filtered_df[column_name].iloc[
            categories_expanded = filtered_df[column_name].str.split('|').explode()
        else:
            categories_expanded = filtered_df[column_name]

        # Шаг 2: Подсчёт вхождений каждой категории
```

```
        category_counts = categories_expanded.value_counts()

        # Шаг 3: Расчёт процентного соотношения каждой категории
        total_categories = category_counts.sum()
        category_percentage = (category_counts / total_categories) * 100

        # Шаг 4: Выбор 10 наиболее часто встречающихся категорий
        top_10_categories_percentage = category_percentage.head(10)

        # Построение гистограммы
        ax = axes[i]
        bars = ax.bar(top_10_categories_percentage.index, top_10_categories_percentage, color
        ax.set_xlabel(column_name.capitalize())
        ax.set_ylabel('Процент')
        title_suffix = " для всех статусов" if not status else f" для статуса {status.capital
        ax.set_title(f'Топ 10 {column_name.capitalize()} по процентному соотношению{title_suf
        ax.set_xticks(range(len(top_10_categories_percentage.index)))
        ax.set_xticklabels(top_10_categories_percentage.index, rotation=45)

        # Добавление текста над каждым столбцом
        for bar in bars:
            yval = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2, yval, f'{yval:.2f}%', ha='center', va='b

        # Вывод первых пяти категорий
        print(f"Топ 10 имен категории в столбце {column_name.capitalize()}{title_suffix}:")
        print(list(top_10_categories_percentage.index[:10]))

    plt.show()
```

In [25]:

```
#Анализ целевого признака
labels = train_df['status'].value_counts().index.tolist()
sizes = train_df['status'].value_counts().tolist()
explode = [0, 0.2]
textprops = {"fontsize":15}

fig, ax = plt.subplots(figsize=(6,6))
ax.set_title('Статус', fontsize=22)
ax.pie(sizes, explode=explode, labels=labels,
       autopct='%1.1f%%', shadow=True, startangle=90,
       radius = 1, textprops =textprops, wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'})
plt.show()

print(train_df['status'].value_counts())
```

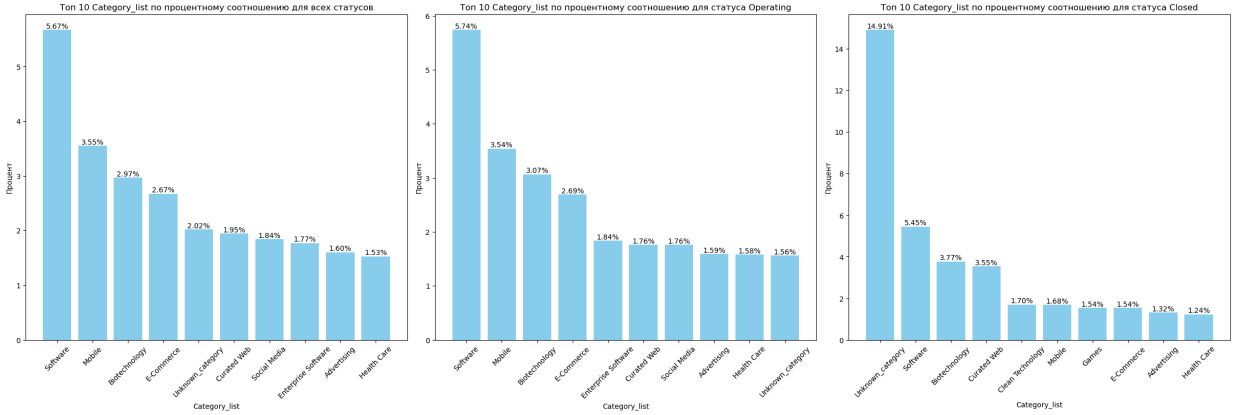



```
operating    47916
closed        4935
Name: status, dtype: int64
```

Присутствует дисбаланс целевого признака

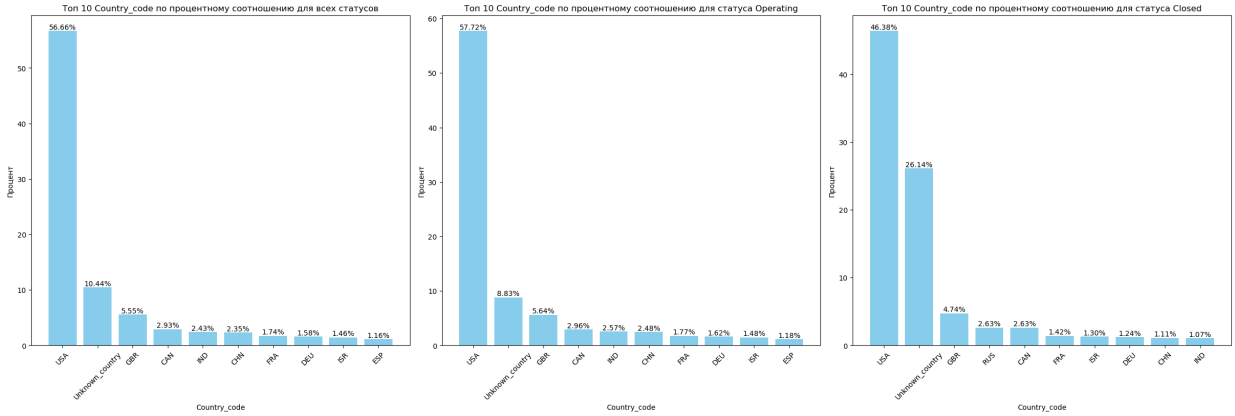
In [26]:
```
plot_column_data('category_list')
```

```
Топ 10 имен категории в столбце Category_list для всех статусов:
['Software', 'Mobile', 'Biotechnology', 'E-Commerce', 'Unknown_category', 'Curated Web', 'Soc
ial Media', 'Enterprise Software', 'Advertising', 'Health Care']
Топ 10 имен категории в столбце Category_list для статуса Operating:
['Software', 'Mobile', 'Biotechnology', 'E-Commerce', 'Enterprise Software', 'Curated Web',
'Social Media', 'Advertising', 'Health Care', 'Unknown_category']
Топ 10 имен категории в столбце Category_list для статуса Closed:
['Unknown_category', 'Software', 'Biotechnology', 'Curated Web', 'Clean Technology', 'Mobil
e', 'Games', 'E-Commerce', 'Advertising', 'Health Care']
```



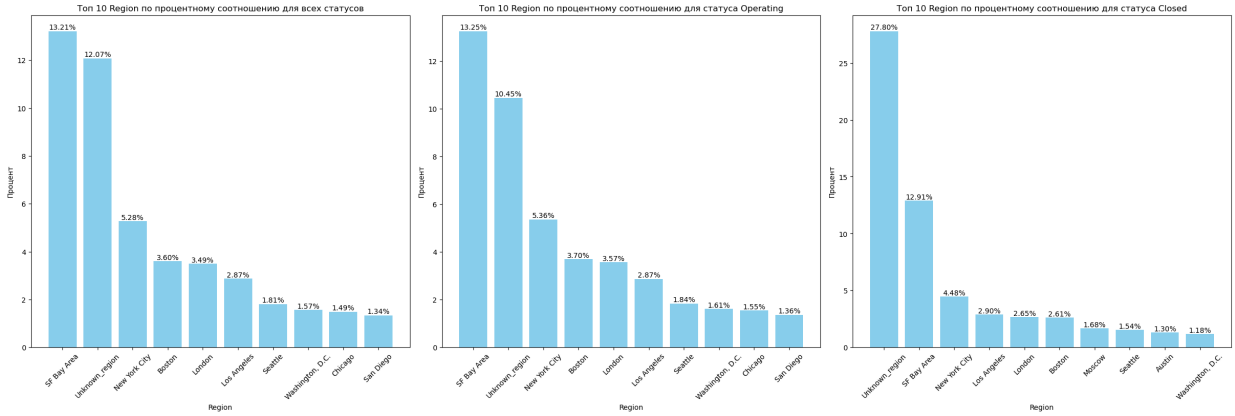
In [27]:
```
plot_column_data('country_code')
```

```
Топ 10 имен категории в столбце Country_code для всех статусов:
['USA', 'Unknown_country', 'GBR', 'CAN', 'IND', 'CHN', 'FRA', 'DEU', 'ISR', 'ESP']
Топ 10 имен категории в столбце Country_code для статуса Operating:
['USA', 'Unknown_country', 'GBR', 'CAN', 'IND', 'CHN', 'FRA', 'DEU', 'ISR', 'ESP']
Топ 10 имен категории в столбце Country_code для статуса Closed:
['USA', 'Unknown_country', 'GBR', 'RUS', 'CAN', 'FRA', 'ISR', 'DEU', 'CHN', 'IND']
```



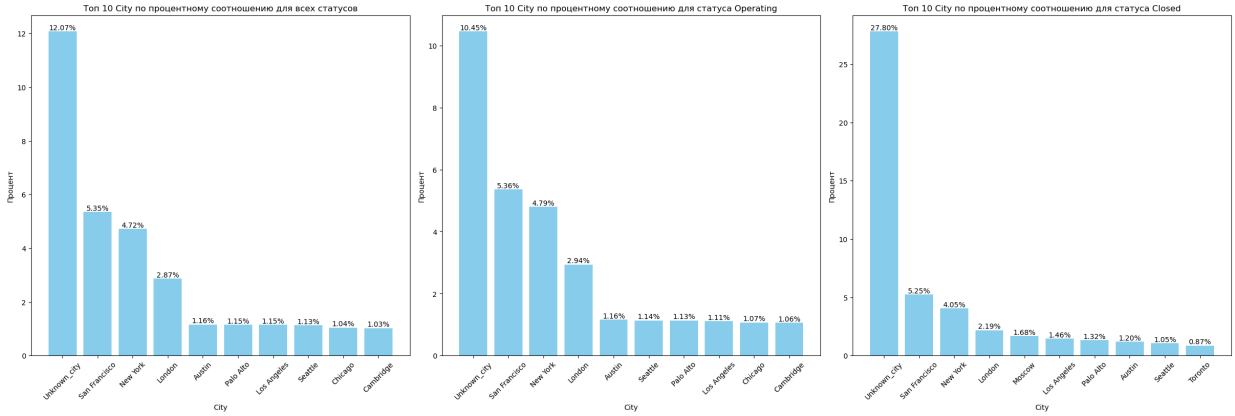
In [28]:
```
plot_column_data('region')
```

```
Топ 10 имен категории в столбце Region для всех статусов:
['SF Bay Area', 'Unknown_region', 'New York City', 'Boston', 'London', 'Los Angeles', 'Seattl
e', 'Washington, D.C.', 'Chicago', 'San Diego']
Топ 10 имен категории в столбце Region для статуса Operating:
['SF Bay Area', 'Unknown_region', 'New York City', 'Boston', 'London', 'Los Angeles', 'Seattl
e', 'Washington, D.C.', 'Chicago', 'San Diego']
Топ 10 имен категории в столбце Region для статуса Closed:
['Unknown_region', 'SF Bay Area', 'New York City', 'Los Angeles', 'London', 'Boston', 'Mosco
w', 'Seattle', 'Austin', 'Washington, D.C.']
```



In [29]:
```
plot_column_data('city')
```

```
Топ 10 имен категории в столбце City для всех статусов:
['Unknown_city', 'San Francisco', 'New York', 'London', 'Austin', 'Palo Alto', 'Los Angeles',
'Seattle', 'Chicago', 'Cambridge']
Топ 10 имен категории в столбце City для статуса Operating:
['Unknown_city', 'San Francisco', 'New York', 'London', 'Austin', 'Seattle', 'Palo Alto', 'Lo
s Angeles', 'Chicago', 'Cambridge']
Топ 10 имен категории в столбце City для статуса Closed:
['Unknown_city', 'San Francisco', 'New York', 'London', 'Moscow', 'Los Angeles', 'Palo Alto',
'Austin', 'Seattle', 'Toronto']
```



- Присутствует дисбаланс целевого признака
- Топ 10 имен категории в столбце Category_list для статуса Closed: ['Unknown_category', 'Software', 'Biotechnology', 'Curated Web', 'Clean Technology', 'Mobile', 'Games', 'E-Commerce', 'Advertising']. Стартапы по которым неизвестна категория закрывались быстрее всех.

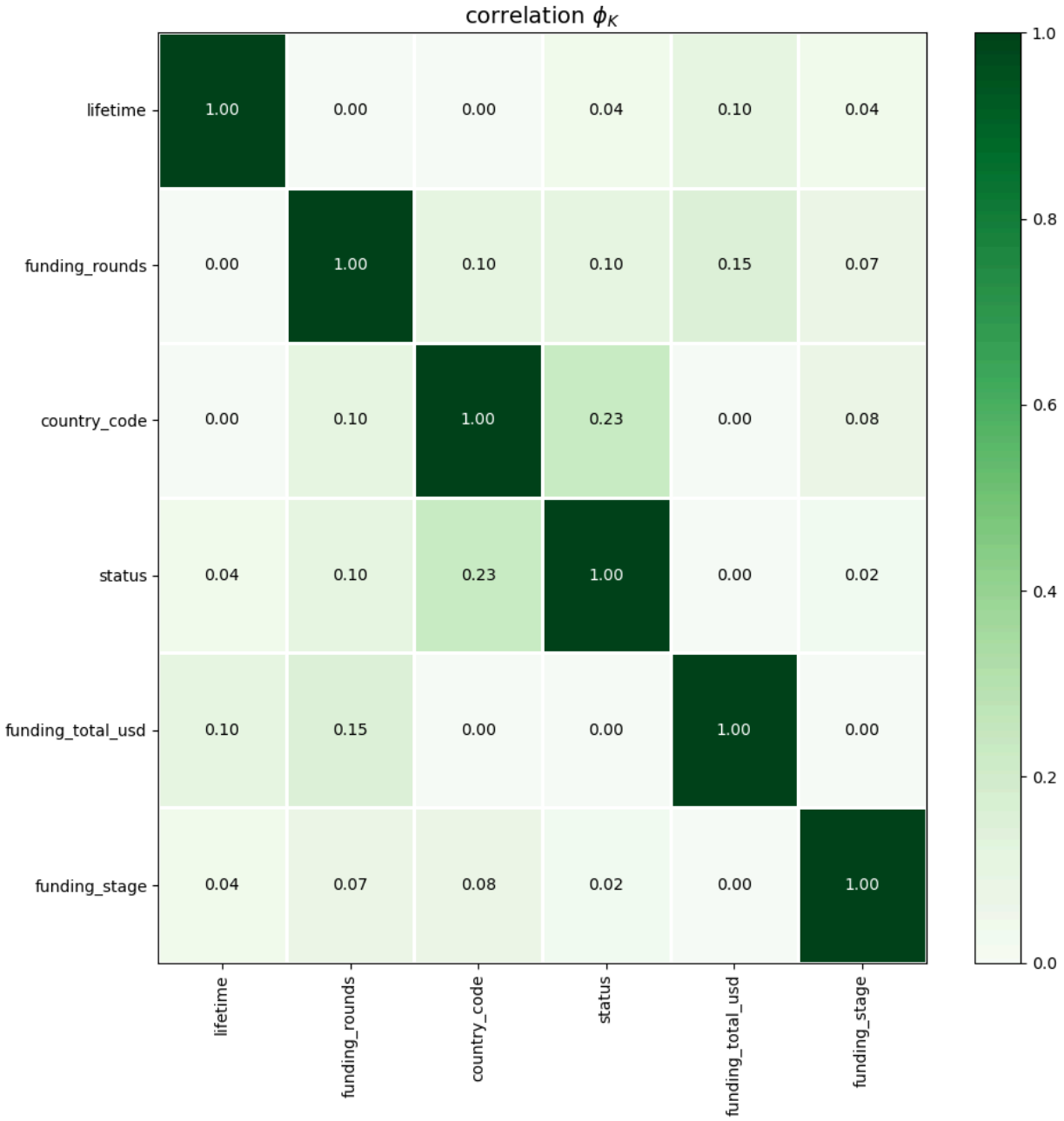
- Топ 10 имен категории в столбце Country_code для статуса Closed: ['USA', 'Unknown_country', 'GBR', 'RUS', 'CAN', 'FRA', 'ISR', 'DEU', 'CHN']. Закрывшиеся стартапы часто находятся в США, возможно просто по причине того, что они там чаще открываются и закрываются.
- Топ 10 имен категории в столбце Region для статуса Closed: ['Unknown_region', 'SF Bay Area', 'New York City', 'Los Angeles', 'London', 'Boston', 'Moscow', 'Seattle', 'Austin']. Стартапы из неизвестного региона чаще закрываются.
- Топ 10 имен категории в столбце City для статуса Closed: ['Unknown_city', 'San Francisco', 'New York', 'London', 'Moscow', 'Los Angeles', 'Palo Alto', 'Austin', 'Seattle']

Таким образом ключевая находка по закрывшимся стартапам - о них было мало информации

Корреляционная матрица

In [30]:
```
# Признаки для содели
columns_to_train = ['funding_stage', 'funding_total_usd', 'status', 'country_code', 'funding_
]
columns_to_test = ['funding_stage', 'funding_total_usd', 'country_code', 'funding_rounds', 'l
]

# Создание новых датафреймов
new_train_df = train_df[columns_to_train].copy()
X_test_final = test_df[columns_to_test].copy()
```

In [31]:
```
from phik.report import plot_correlation_matrix

phik_overview = new_train_df.phik_matrix(interval_cols=['funding_total_usd', 'funding_rounds'
plot_correlation_matrix(phik_overview.values, x_labels=phik_overview.columns, y_labels=phik_o
                        vmin=0, vmax=1, color_map='Greens', title=r'correlation $\phi_K$', fo
                        figsize=(10,10))
plt.tight_layout()
```



Мультиколлинеарность отсутствует. Наибольшая корреляция между раундами финансирования и объемом финансирования.

Заключение по исследованию данных

Анализ количественных данных:

- В среднем 1 раунд финансирования у всех наблюдений
- Присутствует отрицательный lifetime, скорее всего это аномалия
- В среднем между первым и последним финансированием проходит 346 дней
- Присутствуют отрицательные значения времени до первого фандинга перед созданием компании, возможно финансирование поступило на pre seed фазе

Анализ качественных данных:

- Присутствует дисбаланс целевого признака
- Топ 10 имен категории в столбце Category_list для статуса Closed: ['Unknown_category', 'Software', 'Biotechnology', 'Curated Web', 'Clean Technology', 'Mobile', 'Games', 'E-Commerce', 'Advertising']. Стартапы по которым неизвестна категория закрывались быстрее всех.
- Топ 10 имен категории в столбце Country_code для статуса Closed: ['USA', 'Unknown_country', 'GBR', 'RUS', 'CAN', 'FRA', 'ISR', 'DEU', 'CHN']. Закрывшиеся стартапы часто находятся в США, возможно

просто по причине того, что они там чаще открываются и закрываются.

- Топ 10 имен категории в столбце Region для статуса Closed: ['Unknown_region', 'SF Bay Area', 'New York City', 'Los Angeles', 'London', 'Boston', 'Moscow', 'Seattle', 'Austin']. Стартапы из неизвестного региона чаще закрывются.
- Топ 10 имен категории в столбце City для статуса Closed: ['Unknown_city', 'San Francisco', 'New York', 'London', 'Moscow', 'Los Angeles', 'Palo Alto', 'Austin', 'Seattle']

Таким образом ключевая находка по закрывшимся стартапам - о них было мало информации

Мультиколлинеарность отсутствует. Наибольшая корреляция между раундами финансирования и объемом финансирования. В модели будем работать со следующими признаками: 'funding_stage', 'funding_total_usd', 'status', 'country_code', 'funding_rounds', 'lifetime'

Pipeline

In [32]:
```
TEST_SIZE = 0.4
RANDOM_STATE = 42
```

Подготовка данных

In [33]:
```
# Данные для обучения
X = new_train_df.drop(['status'], axis=1)
y = new_train_df['status']
y = y.replace({'operating':0, 'closed':1})

cat_columns = X.select_dtypes(exclude='number').columns.tolist()

# Приведение категориальных признаков к типу category
for col in cat_columns:
    X[col] = X[col].astype('str')

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size = TEST_SIZE,
    random_state = RANDOM_STATE,
    stratify = y)

X_train.shape, X_test.shape
```

Out[33]:
```
((31710, 5), (21141, 5))
```

In [35]:
```
ohe_columns = X_train.select_dtypes(exclude='number').columns.tolist()
num_columns = X_train.select_dtypes(exclude='object').columns.tolist()
ord_columns = ohe_columns.copy()

print('ohe_columns:', ohe_columns)
print('ord_columns:', num_columns)
print('num_columns:', num_columns)
```

```
ohe_columns: ['funding_stage', 'country_code']
ord_columns: ['funding_total_usd', 'funding_rounds', 'lifetime']
num_columns: ['funding_total_usd', 'funding_rounds', 'lifetime']
```

In [36]:
```
# Сбор уникальных категорий для каждого столбца
unique_categories = [X_train[col].dropna().unique().tolist() for col in ord_columns]
print('unique_categories:', unique_categories)
```

```
unique_categories: [['after seed', 'pre-seed'], ['DEU', 'Unknown_country', 'USA', 'CHL', 'CA
N', 'GBR', 'BGR', 'ISR', 'FRA', 'BEL', 'AUS', 'JPN', 'SGP', 'IRL', 'CHE', 'ESP', 'RUS', 'CH
N', 'BRA', 'IND', 'HKG', 'AUT', 'SWE', 'DNK', 'KOR', 'LVA', 'ARG', 'NOR', 'URY', 'KEN', 'HR
V', 'FIN', 'JOR', 'NGA', 'NLD', 'PRT', 'GRC', 'TWN', 'CZE', 'ITA', 'NZL', 'IDN', 'ARE', 'ME
X', 'PER', 'POL', 'EGY', 'THA', 'ZAF', 'EST', 'RWA', 'TUR', 'SVN', 'PHL', 'MYS', 'COL', 'LT
U', 'KWT', 'GHA', 'HUN', 'LBN', 'SAU', 'CYP', 'ISL', 'KHM', 'GIB', 'ECU', 'JEY', 'CYM', 'CM
R', 'UKR', 'SVK', 'LIE', 'ZWE', 'MLT', 'ROM', 'BWA', 'TAN', 'CRI', 'VNM', 'LUX', 'PAK', 'UG
A', 'BLM', 'BLR', 'IRN', 'VEN', 'AZE', 'SRB', 'UZB', 'LKA', 'HND', 'PAN', 'BHR', 'GEO', 'JA
M', 'MNE', 'PRI', 'MDA', 'OMN', 'MCO', 'BMU', 'DOM', 'DZA', 'SYC', 'MUS', 'MKD', 'BAH', 'NP
L', 'KNA', 'MMR', 'ALB', 'SLV', 'SOM', 'BLZ', 'LAO', 'GTM', 'BGD', 'TTO', 'ZMB', 'NIC', 'MA
R', 'ARM', 'QAT', 'TUN', 'MOZ', 'KAZ']]
```

In [37]:

```
# Пайплайн для OneHotEncoding категориальных признаков
ohe_pipe = Pipeline(
    [
        (
            'simpleImputer_ohe',
            SimpleImputer(missing_values=np.nan, strategy='most_frequent')
        ),
        (
            'ohe',
            OneHotEncoder(drop='first', handle_unknown= 'infrequent_if_exist', sparse=False)
        )
    ]
)

# Пайплайн для OrdinalEncoding категориальных признаков
ord_pipe = Pipeline(
    [
        (
            'simpleImputer_ord',
            SimpleImputer(missing_values=np.nan, strategy='most_frequent')
        ),
        (
            'ord',
            OrdinalEncoder(categories=unique_categories,
                           handle_unknown='use_encoded_value', unknown_value=-1)
        )
    ]
)

# Пайплайн для числовых признаков
num_pipe = Pipeline(
    [
        (
            'simpleImputer_num',
            SimpleImputer(missing_values=np.nan, strategy='mean')
        ),
        (
            'scaler',
            StandardScaler()
        )
    ]
)
```

In [38]:

```
# Общий пайплайн для подготовки данных
data_preprocessor = ColumnTransformer(
    [
        ('ohe', ohe_pipe, ohe_columns),
        ('num', num_pipe, num_columns)
    ],
    remainder='passthrough'
)

data_preprocessor_ord = ColumnTransformer(
    [
        ('ord', ord_pipe, ord_columns),
        ('num', num_pipe, num_columns)
    ],
    remainder='passthrough'
)

pipe_final = Pipeline(
    [
        ('preprocessor', 'passthrough'),
        ('models', DecisionTreeClassifier(random_state=RANDOM_STATE))
```

```
    ]
)
```

In [39]:
```
# Определим индексы категориальных признаков для CatBoostClassifier
cat_features_indices = [i for i, col in enumerate(ord_columns)]

# Словарь гиперпараметров моделей
param_dist = [
    # Словарь для модели DecisionTreeClassifier
    {
        'models': [DecisionTreeClassifier(random_state=RANDOM_STATE)],
        'models__max_depth': range(1, 10),
        'models__max_features': range(1, len(X_train.columns) + 1),
        'preprocessor': [data_preprocessor_ord]
    },

    # Словарь для модели KNeighborsClassifier
    {
        'models': [KNeighborsClassifier()],
        'models__n_neighbors': range(1, 10),
        'preprocessor': [data_preprocessor]
    },

    # Словарь для модели LogisticRegression
    {
        'models': [LogisticRegression(
            random_state=RANDOM_STATE,
            solver='saga',
            penalty='l1'
        )],
        'models__C': range(1, 10),
        'preprocessor': [data_preprocessor]
    },

    # Словарь для модели RandomForestClassifier
    {
        'models': [RandomForestClassifier(random_state=RANDOM_STATE)],
        'models__n_estimators': range(50, 200, 50),
        'models__max_depth': range(1, 20),
        'models__max_features': range(1, len(X_train.columns) + 1),
        'models__min_samples_split':[2,5,10],
        'models__min_samples_leaf':[1,2,4],
        'preprocessor': [data_preprocessor_ord]
    },

    # Словарь для модели CatBoostClassifier
    {
        'models': [CatBoostClassifier(random_state=RANDOM_STATE, silent=True, cat_features=oh
        'models__depth': range(1, 10),
        'models__iterations': range(100, 500),
        'preprocessor': ['passthrough']
    }
]

random_search = RandomizedSearchCV(
    pipe_final,
    param_distributions=param_dist,
    cv=5,
    scoring='f1',
    n_jobs=-1,
    n_iter=50,  # Количество итераций для RandomizedSearch
    random_state=RANDOM_STATE
)
```

Обучение и проверка модели

Обучение

In [40]:
```
random_search.fit(X_train, y_train)
```

Out[40]:
```
RandomizedSearchCV(cv=5,
                   estimator=Pipeline(steps=[('preprocessor', 'passthrough'),
                                             ('models',
                                              DecisionTreeClassifier(random_state=42))]),
                   n_iter=50, n_jobs=-1,
                   param_distributions=[{'models': [DecisionTreeClassifier(random_state=42)],
                                         'models__max_depth': range(1, 10),
                                         'models__max_features': range(1, 6),
                                         'preprocessor': [ColumnTransformer(remainder='passth
rough'...

'country_code']),
                                                                                ('n
um',
                                                                                 Pi
peline(steps=[('simpleImputer_num',

SimpleImputer()),

('scaler',

StandardScaler())]),

['funding_total_usd',

'funding_rounds',

'lifetime'])])]},
                                        {'models': [<catboost.core.CatBoostClassifier object
at 0x7f9d00e42970>],
                                         'models__depth': range(1, 10),
                                         'models__iterations': range(100, 500),
                                         'preprocessor': ['passthrough']}],
                   random_state=42, scoring='f1')
```

In [41]:
```
print('Лучшая модель и её параметры:\n\n', random_search.best_estimator_)
print ('Метрика лучшей модели на кросс-валидации:', random_search.best_score_)

#Запишем модель и предпроцессор
model = random_search.best_estimator_.named_steps['models']
preprocessor = random_search.best_estimator_.named_steps['preprocessor']
```

```
Лучшая модель и её параметры:

 Pipeline(steps=[('preprocessor', 'passthrough'),
                 ('models',
                  <catboost.core.CatBoostClassifier object at 0x7f9d157e9160>)])
Метрика лучшей модели на кросс-валидации: 0.7049011433727956
```

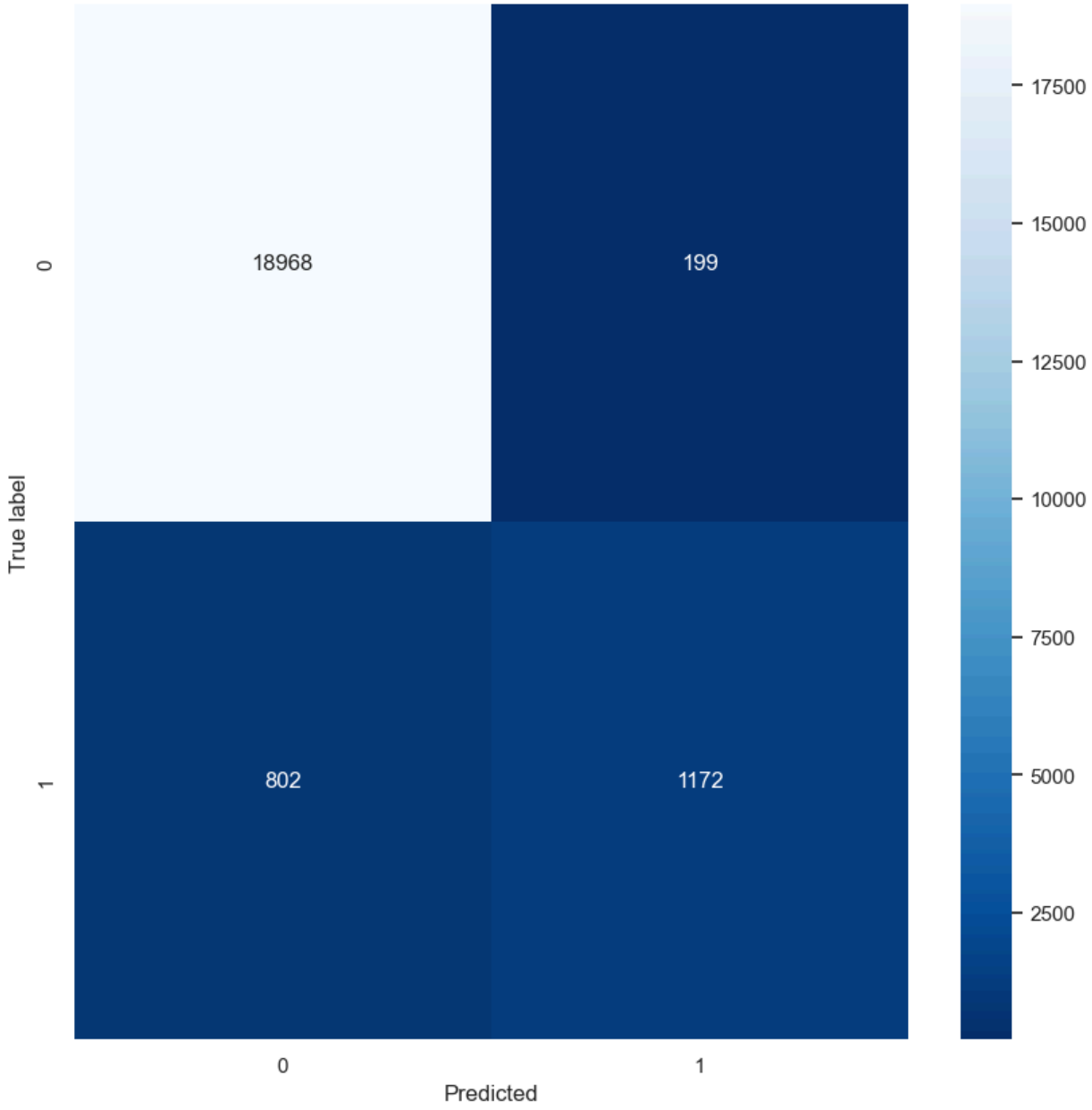
In [42]:
```
pd.set_option('display.max_colwidth', None)

result = pd.DataFrame(random_search.cv_results_)
display(result[
    ['rank_test_score', 'param_models', 'mean_test_score','params']
].sort_values('rank_test_score').head())
```

	rank_test_score	param_models	mean_test_score	params
10	1	<catboost.core.CatBoostClassifier object at 0x7f9d00e42970>	0.704901	{'preprocessor': 'passthrough', 'models__iterations': 250, 'models__depth': 8, 'models': <catboost.core.CatBoostClassifier object at 0x7f9d00e42970>}
1	2	<catboost.core.CatBoostClassifier object at 0x7f9d00e42970>	0.702138	{'preprocessor': 'passthrough', 'models__iterations': 462, 'models__depth': 7, 'models': <catboost.core.CatBoostClassifier object at 0x7f9d00e42970>}
11	3	<catboost.core.CatBoostClassifier object at 0x7f9d00e42970>	0.701507	{'preprocessor': 'passthrough', 'models__iterations': 116, 'models__depth': 3, 'models': <catboost.core.CatBoostClassifier object at 0x7f9d00e42970>}
3	4	<catboost.core.CatBoostClassifier object at 0x7f9d00e42970>	0.701470	{'preprocessor': 'passthrough', 'models__iterations': 263, 'models__depth': 7, 'models': <catboost.core.CatBoostClassifier object at 0x7f9d00e42970>}
2	5	<catboost.core.CatBoostClassifier object at 0x7f9d00e42970>	0.701402	{'preprocessor': 'passthrough', 'models__iterations': 298, 'models__depth': 7, 'models': <catboost.core.CatBoostClassifier object at 0x7f9d00e42970>}

In [43]:
```
y_pred = random_search.predict(X_test)
print(f"F1 = {f1_score(y_test, y_pred, pos_label=1):.2f}")
```

```
F1 = 0.70
```

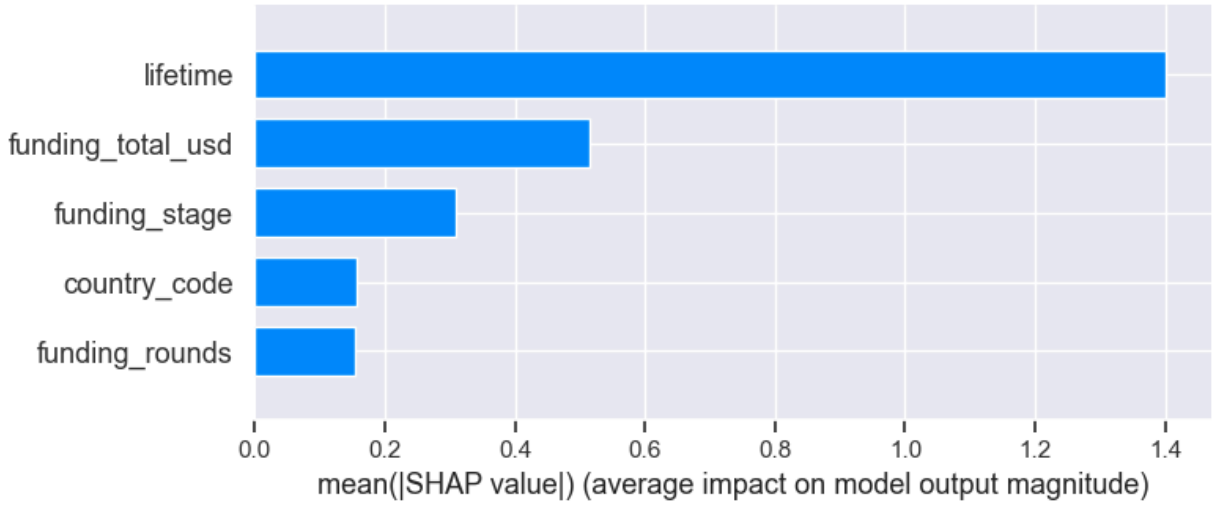
In [44]:
```
cm = confusion_matrix(y_test, y_pred)
sns.set(rc={'figure.figsize':(10,10)})
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues_r')
plt.ylabel('True label')
plt.xlabel('Predicted');
```



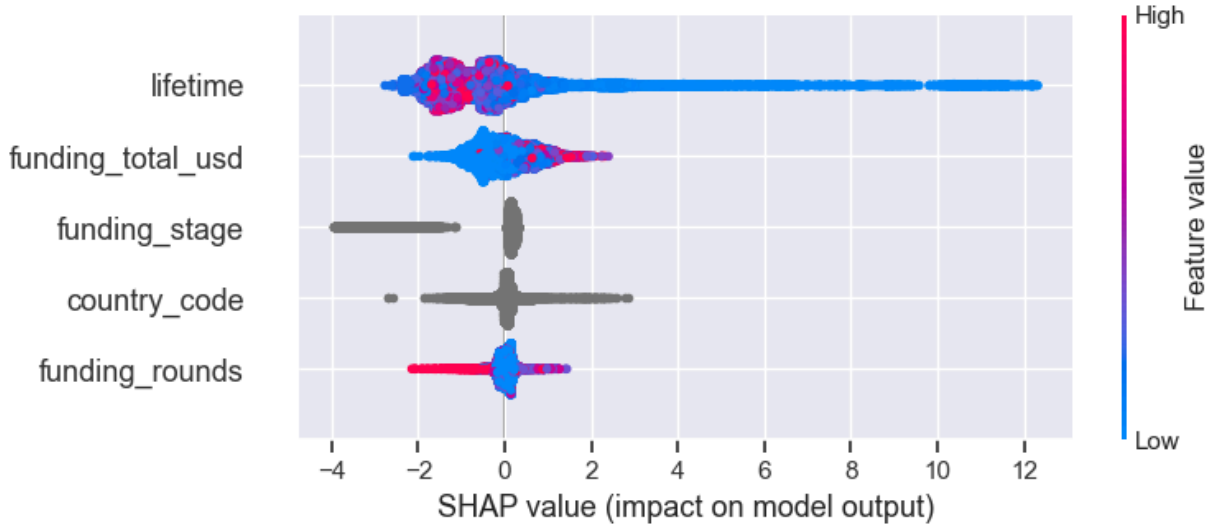
False positive ошибки для нас болезнены тем, что мы не найдем гем среди стартапов. Как венчурные инвесторы мы вкладываемся во многие стартапы но по чуть-чуть.

Анализ важности признаков

In [45]:
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_train)

shap.summary_plot(shap_values, X_train, plot_type="bar", max_display=30)
```



In [46]:
```
feature_names = X_test.columns

# Инициализация explainer с параметром feature_perturbation
explainer = shap.TreeExplainer(random_search.best_estimator_.named_steps['models'], feature_p

# Получение SHAP значений
shap_values = explainer.shap_values(X_test)

# Преобразование в объект Explanation
shap_values_exp = shap.Explanation(values=shap_values, base_values=explainer.expected_value,

# Визуализация
shap.summary_plot(shap_values_exp, X_test)
```



Вывод

Наилучшая модель:

- Лучшая модель и её параметры: CatBoostClassifier
- Метрика лучшей модели на кросс-валидации: 0.7049011433727956
- lifetime наиболее важный признак

Заключение

Выводы

Заключение по загрузке и ознакомлению с данными:

- Тренировочные данные представлены таблицей размерностью 52879 строк и 13 столбцов
- Тестовые данные представлены 13211 строк и 12 столбцов
- Содержание столбцов схоже, в тестовых данных отсутствует столбец с целевым признаком

Столбцы:

- **name** - Название стартапа
- **category_list** - Индустрия в которой работает стартап
- **funding_total_usd** - Объем инвестиций
- **status** - статус функционирования (целевой признак, отсутствует в тестовой выборке)
- **country_code** - страна
- **state_code** - штат
- **region** - Регион стартапа
- **city** - Город где функционирует стартап
- **funding_rounds** - число раундов финансирования
- **founded_at** - дата создания стартапа
- **first_funding_at** - дата первого раунда финансирования
- **last_funding_at** - дата последнего раунда финансирования
- **closed_at** - дата закрытия стартапа

Заключение по предподготовке данных:

- Тренировочный датасет и тестовый датасет имеют схожую статистику по пропускам
- В тренировочном файле дубликаты не выявлены
- 91% наблюдений не содержит информацию о том, когда закрылись компании, а следовательно предполагаем, что они еще работают.
- 5% наблюдений не содержат информацию в каком рынке функционирует стартап
- 10% не содержит информацию о стране происхождения
- 13% о штате происхождения
- 12% не содержин информацию о регионе происхождения
- 12% не содержит информацию о городе происхождения
- Отсутствующие данные заменены на заглушки

Созданы новые признаки на базе столбцов с датами:

- lifetime продолжительность жизни стартапа от даты создания до даты закрытия
- funding_length продолжительность жизни стартапа между первым и последним раундом финансирования
- first_funding_lt продолжительность жизни стратапа между первым финансированием и созданием
- funding_life_share - доля жизни стартапа с финансированием

Анализ количественных данных:

- В среднем 1 раунд финансирования у всех наблюдений
- Присутствует отрицательный lifetime, скорее всего это аномалия
- В среднем между первым и последним финансированием проходит 346 дней
- Присутствуют отрицательные значения времени до первого фандинга перед созданием компании, возможно финансирование поступило на pre seed фазе

Анализ качественных данных:

- Присутствует дисбаланс целевого признака
- Топ 10 имен категории в столбце Category_list для статуса Closed: ['Unknown_category', 'Software', 'Biotechnology', 'Curated Web', 'Clean Technology', 'Mobile', 'Games', 'E-Commerce', 'Advertising']. Стартапы по которым неизвестна категория закрывались быстрее всех.
- Топ 10 имен категории в столбце Country_code для статуса Closed: ['USA', 'Unknown_country', 'GBR', 'RUS', 'CAN', 'FRA', 'ISR', 'DEU', 'CHN']. Закрывшиеся стартапы часто находятся в США, возможно просто по причине того, что они там чаще открываются и закрываются.

- Топ 10 имен категории в столбце Region для статуса Closed: ['Unknown_region', 'SF Bay Area', 'New York City', 'Los Angeles', 'London', 'Boston', 'Moscow', 'Seattle', 'Austin']. Стартапы из неизвестного региона чаще закрывются.
- Топ 10 имен категории в столбце City для статуса Closed: ['Unknown_city', 'San Francisco', 'New York', 'London', 'Moscow', 'Los Angeles', 'Palo Alto', 'Austin', 'Seattle']

Таким образом ключевая находка по закрывшимся стартапам - о них было мало информации

- Мультиколлинеарность отсутствует. Наибольшая корреляция между раундами финансирования и объемом финансирования.
- В модели будем работаем со следующими признаками: 'funding_stage', 'funding_total_usd', 'country_code', 'funding_rounds', 'lifetime'

Наилучшая модель:

- Сравнивались модели RandomForrestClassifier, DecissionTreeClassifier, LogisticRegression, KNeighborsClassifier
- Лучшая модель и её параметры: CatBoostClassifier
- Метрика лучшей модели на кросс-валидации: 0.7049011433727956
- lifetime наиболее важный признак

Предложения по улучшению модели и применение

- Для венчурных инвесторов можно продолжить развитие модели включив анализ порога классификации.
- Стоит обратить на качество инвесторов осуществивших инвестиции.
- Можно изучить параметры фаундеров, включить информацию о количестве кризисов которые смогла пережить компания