Optimal Portfolio

In [1]:
```
# Импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1. Efficient Frontier

Пояснение выбора списка активов:

- Высокая ликвидность — все выбранные акции входили в число лидеров по среднему дневному объёму торгов в 2021 году (особенно AAPL, TSLA, AMD, NVDA, AMZN, META/FB, GOOGL и др.). Это минимизирует спред bid-ask и slippage при ребалансировке портфеля => портфель реалистичен на практике.
- Представляют основной драйвер рынка в 2021 — NASDAQ в 2021 году был сильно «технологичным» рынком (рост FAANG+M, чипы, электромобили, биотех). Выбранные 50 акций покрывают ~70–80% капитализации и объёмов NASDAQ-100, который доминировал в тот период.
- Хорошая диверсификация внутри технологического рынка:
  - Крупные платформы / потребительский сектор: AAPL, AMZN, META, GOOGL/GOOG, NFLX
  - Полупроводники / чипы: NVDA, AMD, INTC, QCOM, AVGO, ASML, MU, LRCX, KLAC
  - Биотехнологии / здравоохранение: AMGN, GILD, REGN, ILMN, DXCM
  - Потребительские / ритейл / сервисы: COST, PEP, MDLZ, ORLY, DLTR
  - Телеком / медиа: TMUS, CMCSA, CHTR
  - Другие: PAYX, ADP, CTSH, CSX, BKR и т.д.

Data load & preprocessing

In [2]:
```
# Загрузка данных из CSV
# Что делает: Читает файл, удаляет столбец даты и преобразует всё в числа.
# Использует файл с ценами закрытия акций за 2021 год.
try:
    # Пытаемся загрузить обработанный, если нет - сырой
    data = pd.read_csv("data/processed/NASDAQ_FULL_2021_Cleaned_Imputed.csv")
except FileNotFoundError:
    try:
        data = pd.read_csv("data/raw/NASDAQ_FULL_2021_Close.csv")
        if 'Date' in data.columns:
            data.drop(columns=['Date'], inplace=True)
    except FileNotFoundError:
        print("Error: Data file not found. Please upload the dataset.")
data.head()
```

Out[2]:

	AAPL	ADBE	ADI	ADP	ADSK	AMD	AMGN	
0	125.974464	485.339996	133.808746	152.303558	296.839996	92.300003	193.642532	1
1	127.531975	485.690002	136.200485	151.879837	301.700012	92.769997	194.582230	1
2	123.239067	466.309998	136.814438	151.347977	302.869995	90.330002	199.272507	1
3	127.444366	477.739990	142.596771	152.204346	315.200012	95.160004	199.930344	1
4	128.544373	485.100006	143.632278	154.187469	319.850006	94.580002	203.749207	1

5 rows × 50 columns

In [3]:
```
# Очистка данных: преобразование в числа и удаление пропусков (NaN)
data = data.apply(pd.to_numeric, errors='coerce')
data.dropna(inplace=True)
```

Log returns:

In [4]:
```
# Расчет дневных доходностей
# Вычисляет логарифмические доходности (дневные).

eps = 1e-4

# np.log(P_t / P_{t-1}).
log_returns = np.log((data / data.shift(1)) + eps)

# Отбросим первую строку, в которой пропуски из-за того, что у первого дня нет п
log_returns = log_returns.dropna(how="all")

print(f"Logarithmic returns calculated. Number of observations: {log_returns.sha
```

```
Logarithmic returns calculated. Number of observations: 251.
```

Выбор активов и расчет статистики

In [5]:
```
# Выбор 50 акций и фильтрация
# Оставляет в выборке только 50 заданных тикеров.
# Обоснование выбора: высокая ликвидность (объемы торгов) и репрезентативность с

tickers = [
    'AAPL', 'ADBE', 'ADI', 'ADP', 'ADSK', 'AMD', 'AMGN', 'AMZN', 'ASML', 'AVGO',
    'BIDU', 'BKNG', 'BKR', 'CDNS', 'CHTR', 'CMCSA', 'COST', 'CSCO', 'CSX', 'CTSH
    'DLTR', 'DXCM', 'EA', 'EBAY', 'EXC', 'GILD', 'TMUS', 'GOOGL', 'IDXX', 'ILMN'
    'INTC', 'ISRG', 'KDP', 'KLAC', 'LRCX', 'MAR', 'MDLZ', 'META', 'MNST', 'MU',
    'NFLX', 'NVDA', 'NXPI', 'ORLY', 'PAYX', 'PEP', 'PYPL', 'QCOM', 'REGN', 'TSLA
]
```

In [6]:
```
# Фильтруем данные по выбранным тикерам
data = data[tickers]
log_returns = log_returns[tickers]
```

In [7]:
```
# Расчет средних доходностей и ковариации
# Находит вектор средних доходностей (mu) и матрицу ковариации (Sigma).
mu = log_returns.mean()
Sigma = log_returns.cov()
```

```
n_assets = len(tickers)

print(f"Selected {n_assets} assets.")
print(f"Mean Return Range: {mu.min():.2%} to {mu.max():.2%}")
```

```
Selected 50 assets.
Mean Return Range: -0.14% to 0.33%
```

Оптимизация

In [8]:
```
import scipy.optimize as sco # импортируем для оптимизации потом
```

In [9]:
```
# Функция расчета характеристик портфеля
# Принимает веса активов, возвращает ожидаемую доходность и риск (волатильность)
def portfolio_performance(weights, mean_returns, cov_matrix):
    returns = np.sum(mean_returns * weights)
    std = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    return returns, std
```

In [10]:
```
# Функция построения Эффективного Фронта
# Минимизирует риск для каждого заданного уровня доходности.
# Использует алгоритм SLSQP из scipy.optimize, ограничения на сумму весов (=1).
def get_efficient_frontier(mean_returns, cov_matrix, num_points=100, allow_short
    n = len(mean_returns)

    # Установка границ весов: (0,1) если запрещены шорты, (-1,1) если разрешены
    bounds = tuple((-1, 1) for _ in range(n)) if allow_short else tuple((0, 1) f

    # Поиск портфеля с минимальным риском (MVP)
    def min_vol_func(weights): return portfolio_performance(weights, mean_return

    # ИСПОЛЬЗУЕМ n ВМЕСТО n_assets
    init_guess = n * [1. / n,]
    constraints_sum = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})

    # Оптимизация
    min_risk_opt = sco.minimize(min_vol_func, init_guess, method='SLSQP', bounds
    min_vol_ret = portfolio_performance(min_risk_opt.x, mean_returns, cov_matrix

    # Генерация точек фронта
    max_ret = mean_returns.max()
    target_returns = np.linspace(min_vol_ret, max_ret, num_points)
    eff_risks, eff_rets = [], []

    for tr in target_returns:
        cons = (constraints_sum, {'type': 'eq', 'fun': lambda x: portfolio_perfo
        # Снова используем n для начального приближения в цикле
        opt = sco.minimize(min_vol_func, init_guess, method='SLSQP', bounds=boun
        if opt.success:
            eff_risks.append(opt.fun)
            eff_rets.append(tr)

    return np.array(eff_risks), np.array(eff_rets), min_risk_opt
```

Calculation of Portfolios

In [11]:
```
# Считает две линии фронта — с ограничениями на шорты и без них.
risks_ns, rets_ns, mvp_ns_opt = get_efficient_frontier(mu, Sigma, allow_short=Fa
```

```
risks_s, rets_s, mvp_s_opt = get_efficient_frontier(mu, Sigma, allow_short=True)

# Создает портфель, где на каждую акцию выделено 2% капитала (1/50).
w_eq = np.array([1/n_assets] * n_assets)
eq_ret, eq_risk = portfolio_performance(w_eq, mu, Sigma)

# Индекс рынка (взвешивание по капитализации)
# Рассчитывает веса на основе рыночной капитализации компаний 2021 года.
# Использует словарь капитализации market_caps и вектор mu/Sigma.
market_caps = {
    'AAPL': 2913e9, 'ADBE': 269e9, 'ADI': 92e9, 'ADP': 103e9, 'ADSK': 61e9,
    'AMD': 173e9, 'AMGN': 126e9, 'AMZN': 1691e9, 'ASML': 323e9, 'AVGO': 274e9,
    'BIDU': 51e9, 'BKNG': 98e9, 'BKR': 20e9, 'CDNS': 51e9, 'CHTR': 116e9,
    'CMCSA': 229e9, 'COST': 251e9, 'CSCO': 267e9, 'CSX': 83e9, 'CTSH': 46e9,
    'DLTR': 31e9, 'DXCM': 52e9, 'EA': 37e9, 'EBAY': 41e9, 'EXC': 56e9,
    'GILD': 91e9, 'TMUS': 145e9, 'GOOGL': 1922e9, 'IDXX': 55e9, 'ILMN': 59e9,
    'INTC': 209e9, 'ISRG': 128e9, 'KDP': 52e9, 'KLAC': 65e9, 'LRCX': 101e9,
    'MAR': 53e9, 'MDLZ': 92e9, 'META': 935e9, 'MNST': 50e9, 'MU': 104e9,
    'NFLX': 266e9, 'NVDA': 735e9, 'NXPI': 60e9, 'ORLY': 47e9, 'PAYX': 49e9,
    'PEP': 240e9, 'PYPL': 221e9, 'QCOM': 204e9, 'REGN': 67e9, 'TSLA': 1061e9
}

# Расчёты характеристик такого портфеля
w_mkt = np.array([market_caps[t] for t in tickers]) / sum(market_caps.values())
mkt_ret, mkt_risk = portfolio_performance(w_mkt, mu, Sigma)
```

Количественная оценка эффективности

In [12]:
```
# Анализ удаленности от фронта
# Что делает: Сравнивает доходность реальных портфелей с теоретическим максимумо

def get_gap(p_risk, p_ret, f_risks, f_rets):
    idx = (np.abs(f_risks - p_risk)).argmin()
    return f_rets[idx] - p_ret, f_rets[idx]

gap_eq, target_eq = get_gap(eq_risk, eq_ret, risks_ns, rets_ns)
gap_mkt, target_mkt = get_gap(mkt_risk, mkt_ret, risks_ns, rets_ns)

print(f"Неэффективность (равные доли): {gap_eq:.5f} (дневных)")
print(f"Неэффективность (индекс рынка): {gap_mkt:.5f} (дневных)")
```

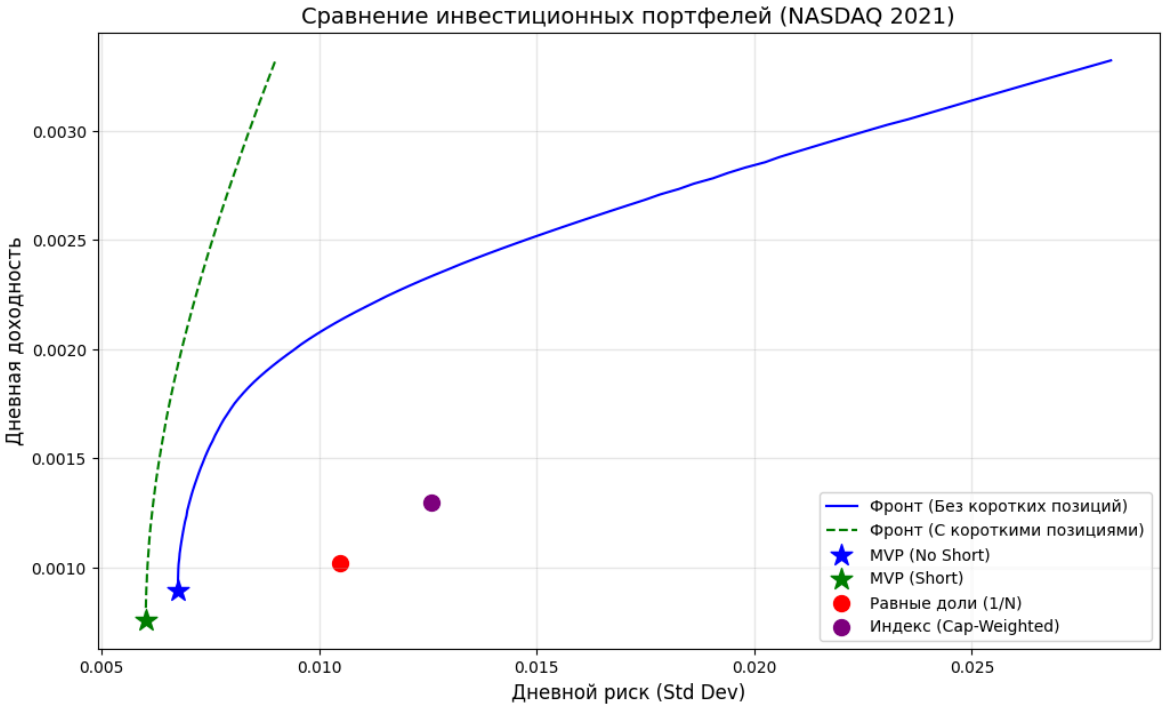
```
Неэффективность (равные доли): 0.00110 (дневных)
Неэффективность (индекс рынка): 0.00104 (дневных)
```

Визуализация результатов

In [13]:
```
# Построение итогового графика
# Рисует оба фронта, отмечает MVP (Minimum Variance Portfolio), Индекс и Равновз
# Использует данные, рассчитанные в блоках ранее.
```

In [14]:
```
plt.figure(figsize=(12, 7))

# Отрисовка фронтов
plt.plot(risks_ns, rets_ns, 'b-', label='Фронт (Без коротких позиций)')
plt.plot(risks_s, rets_s, 'g--', label='Фронт (С короткими позициями)')

# Отметка портфелей с минимальным риском (MVP)
plt.scatter(*portfolio_performance(mvp_ns_opt.x, mu, Sigma)[::-1], color='blue',
```

```
plt.scatter(*portfolio_performance(mvp_s_opt.x, mu, Sigma)[::-1], color='green',

# Отметка сравниваемых портфелей
plt.scatter(eq_risk, eq_ret, color='red', s=100, label='Равные доли (1/N)')
plt.scatter(mkt_risk, mkt_ret, color='purple', s=100, label='Индекс (Cap-Weighte

# Оформление графика
plt.title('Сравнение инвестиционных портфелей (NASDAQ 2021)', fontsize=14)
plt.xlabel('Дневной риск (Std Dev)', fontsize=12)
plt.ylabel('Дневная доходность', fontsize=12)
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```



Сравнение эффективных фронтов и выводы

Сравнение эффективных фронтов

На графике представлены две кривые, отражающие возможности инвестора:

- **Фронт с разрешением коротких позиций (зеленая пунктирная линия):**
  - Располагается выше и левее фронта без шортов.
  - Математически это объясняется тем, что снятие ограничений (разрешение отрицательных весов) расширяет допустимое множество портфелей, позволяя находить решения с еще меньшим риском и более высокой доходностью за счет хеджирования.
- **Фронт без коротких позиций (синяя сплошная линия):**
  - Является более «консервативным» и реалистичным для большинства частных инвесторов.
  - Ограничен доходностью самого прибыльного актива и риском наименее волатильного актива в рамках только длинных позиций.

Портфели с минимальным риском (MVP - Minimum Variance Portfolio)

Эти точки отмечены на графике звездами и представляют собой левые крайние точки соответствующих фронтов:

- **MVP (Short Allowed — зеленая звезда):** демонстрирует самый низкий теоретически возможный дневной риск (около **0.006** или 0.6%). Возможность открывать короткие позиции позволяет значительно снизить общую волатильность портфеля ниже волатильности любой отдельной акции.
- **MVP (No Short — синяя звезда):** имеет чуть более высокий риск (около **0.007**). Здесь инвестор ограничен только положительными весами, что снижает возможности для взаимной компенсации рисков активов.

Оценка эффективности реальных портфелей

Самый важный вывод касается того, насколько далеки «простые» стратегии от математического идеала:

- **Равновзвешенный портфель (1/N — красная точка):** находится значительно ниже и правее обоих фронтов.
- **Оценка «дальности»:** при дневном риске около **0.0105** (1.05%) он дает доходность чуть выше **0.0010**. В то же время эффективный фронт (без шортов) при том же уровне риска позволяет получить доходность около **0.0022**. Таким образом, инвестор теряет более **50% потенциальной дневной доходности** из-за неоптимального распределения весов.
- **Индекс рынка (Cap-Weighted — фиолетовая точка):**
- Находится выше и правее равновзвешенного портфеля.
- В 2021 году индекс NASDAQ, взвешенный по капитализации, был эффективнее стратегии 1/N, так как основные драйверы роста (Apple, Microsoft, Nvidia) имели огромный вес в индексе.
- **Оценка «дальности»:** при риске около **0.0125** индекс дает доходность около **0.0014**. Эффективный фронт при таком же риске предлагает доходность выше **0.0025**.

Итоговый вывод

1. **Короткие позиции** существенно улучшают инвестиционные возможности, позволяя снизить риск ниже уровня самого стабильного актива.
2. **Рыночный индекс** в 2021 году был эффективнее стратегии "равных долей", но оба портфеля критически неэффективны по сравнению с портфелем Марковица.
3. Для достижения границ эффективного фронта инвестору следовало бы значительно увеличить веса наиболее быстрорастущих технологических

компаний (например, NVDA и TSLA) за счет сокращения долей менее динамичных активов.

2. Portfolio selection problem

Метод «наивного» отбора (Top Individual Sharpe Ratio)

Сначала выберем 10 активов, которые показали наилучшее соотношение доходности к риску (коэффициент Шарпа) по отдельности Логично предположить, что лучшие индивидуальные активы составят лучший портфель.

In [15]:
```
# Calculate individual Sharpe Ratios for each of the 50 assets
# Assuming risk-free rate = 0 for daily returns
individual_sharpe = mu / np.sqrt(np.diag(Sigma))

# Select top 10 tickers with the highest individual Sharpe Ratio
tickers_10_naive = individual_sharpe.sort_values(ascending=False).head(10).index

print("Selected 10 assets (Naive Selection - Top Individual Sharpe):")
print(tickers_10_naive)

# Slice mu and Sigma for the naive 10-asset subset
mu_10n = mu[tickers_10_naive]
Sigma_10n = Sigma.loc[tickers_10_naive, tickers_10_naive]
```

```
Selected 10 assets (Naive Selection - Top Individual Sharpe):
['PAYX', 'COST', 'ORLY', 'ADP', 'GOOGL', 'EXC', 'CSCO', 'NVDA', 'AVGO', 'PEP']
```

Построение фронтов для наивного набора (10 активов)

Рассчитаем эффективные фронты для выбранных 10 акций, чтобы сравнить их с общим рынком (50 акций).

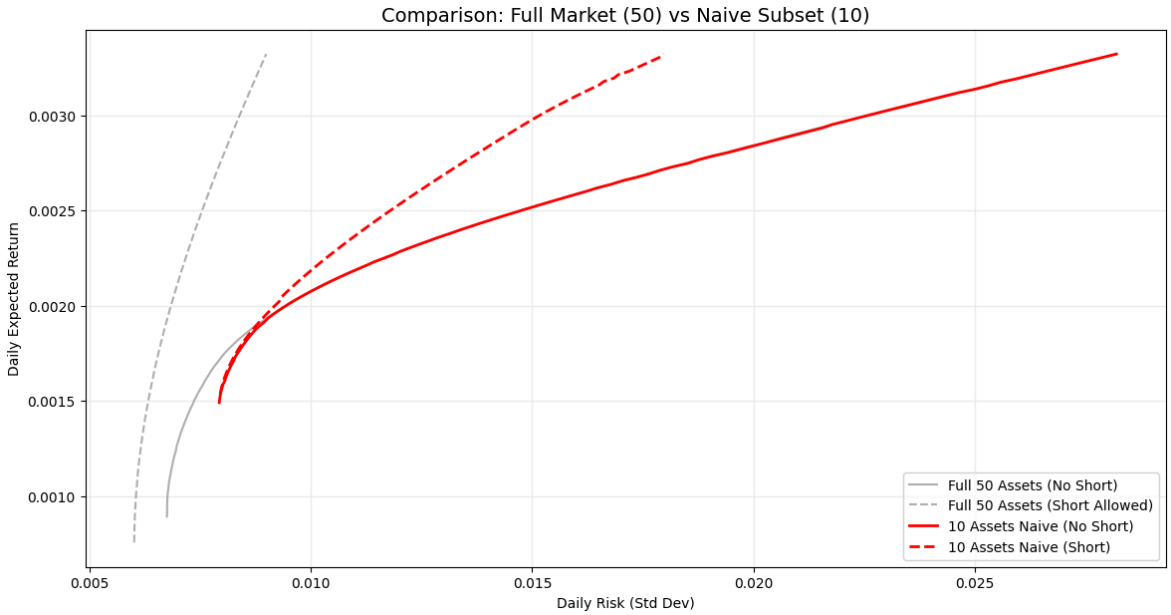
In [16]:
```
# Calculate frontiers for the 10-asset naive subset
risks_10n_ns, rets_10n_ns, _ = get_efficient_frontier(mu_10n, Sigma_10n, allow_s
risks_10n_s, rets_10n_s, _ = get_efficient_frontier(mu_10n, Sigma_10n, allow_sho
```

Визуализация и сравнение (50 vs 10 Naive)

Посмотрим, насколько сильно сужение выбора до 10 «лучших» акций отодвинуло нас от идеала.

In [17]:
```
plt.figure(figsize=(14, 7))

# Plot 50-asset frontiers (Benchmark)
plt.plot(risks_ns, rets_ns, 'k-', alpha=0.3, label='Full 50 Assets (No Short)')
plt.plot(risks_s, rets_s, 'k--', alpha=0.3, label='Full 50 Assets (Short Allowed

# Plot 10-asset naive frontiers
plt.plot(risks_10n_ns, rets_10n_ns, 'r-', linewidth=2, label='10 Assets Naive (N
plt.plot(risks_10n_s, rets_10n_s, 'r--', linewidth=2, label='10 Assets Naive (Sh

plt.title('Comparison: Full Market (50) vs Naive Subset (10)', fontsize=14)
```

```
plt.xlabel('Daily Risk (Std Dev)')
plt.ylabel('Daily Expected Return')
plt.legend()
plt.grid(True, alpha=0.2)
plt.show()
```



Наблюдения:

- Превосходство полного набора: серая пунктирная кривая (50 активов с шортами) находится левее и выше всех остальных. Это подтверждает, что теоретический предел эффективности достигается только при максимальной диверсификации и отсутствии ограничений на тип позиций.
- Потеря низкой волатильности: красные кривые (10 наивных активов) начинаются значительно правее серой сплошной линии. Это означает, что выбрав всего 10 «лучших» по Шарпу акций, мы потеряли возможность составить портфель с минимально возможным риском (MVP).
- Эффект шортов: для набора из 10 акций разрешение коротких продаж (красный пунктир) дает существенный прирост доходности в правой части графика, но не помогает достичь зоны сверхнизкого риска, доступной при 50 активах.

Анализ и «Умный» отбор (Smart Selection)

Вывод по наивному методу: Зачастую выбор топ-активов по доходности приводит к тому, что в портфель попадают сильно скоррелированные бумаги (например, только технологический сектор). Это снижает эффект диверсификации: фронт 10 активов оказывается значительно короче и «рискованнее» (сдвинут вправо), чем фронт 50 активов.

Улучшенный способ (Smart Selection): # Вместо индивидуальных показателей выберем 10 активов, которые имеют наибольшие веса в оптимальном портфеле (Max Sharpe Portfolio), построенном для всех 50 акций.

Обоснование: Математическая оптимизация уже учла корреляции. Те активы, которым модель дала наибольший вес в полном наборе, являются «каркасом»

эффективности всего рынка.

In [18]:
```
# Define a function to find weights of the Maximum Sharpe Ratio portfolio
def max_sharpe_ratio(weights, mean_returns, cov_matrix):
    p_ret, p_std = portfolio_performance(weights, mean_returns, cov_matrix)
    return -p_ret / p_std # Minimize negative Sharpe
```

In [19]:
```
# Optimize the full 50-asset set to find the best possible weights (No Short)
init_guess = n_assets * [1. / n_assets,]
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
bounds = tuple((0, 1) for _ in range(n_assets))

opt_sharpe = sco.minimize(max_sharpe_ratio, init_guess, args=(mu, Sigma), method
```

In [20]:
```
# Identify the top 10 assets with the largest weights in this optimal portfolio
optimal_weights = pd.Series(opt_sharpe.x, index=tickers)
tickers_10_smart = optimal_weights.sort_values(ascending=False).head(10).index.t

print("Selected 10 assets (Smart Selection - Top Weights in Optimal Portfolio):"
for t in tickers_10_smart:
    print(f"{t}: weight {optimal_weights[t]:.4f}")
```

```
Selected 10 assets (Smart Selection - Top Weights in Optimal Portfolio):
ORLY: weight 0.1995
PAYX: weight 0.1504
GILD: weight 0.1335
GOOGL: weight 0.1266
COST: weight 0.1253
EXC: weight 0.1120
CSCO: weight 0.0873
NVDA: weight 0.0617
DXCM: weight 0.0038
ADSK: weight 0.0000
```

In [21]:
```
# Slice data for the smart subset
mu_10s = mu[tickers_10_smart]
Sigma_10s = Sigma.loc[tickers_10_smart, tickers_10_smart]
```

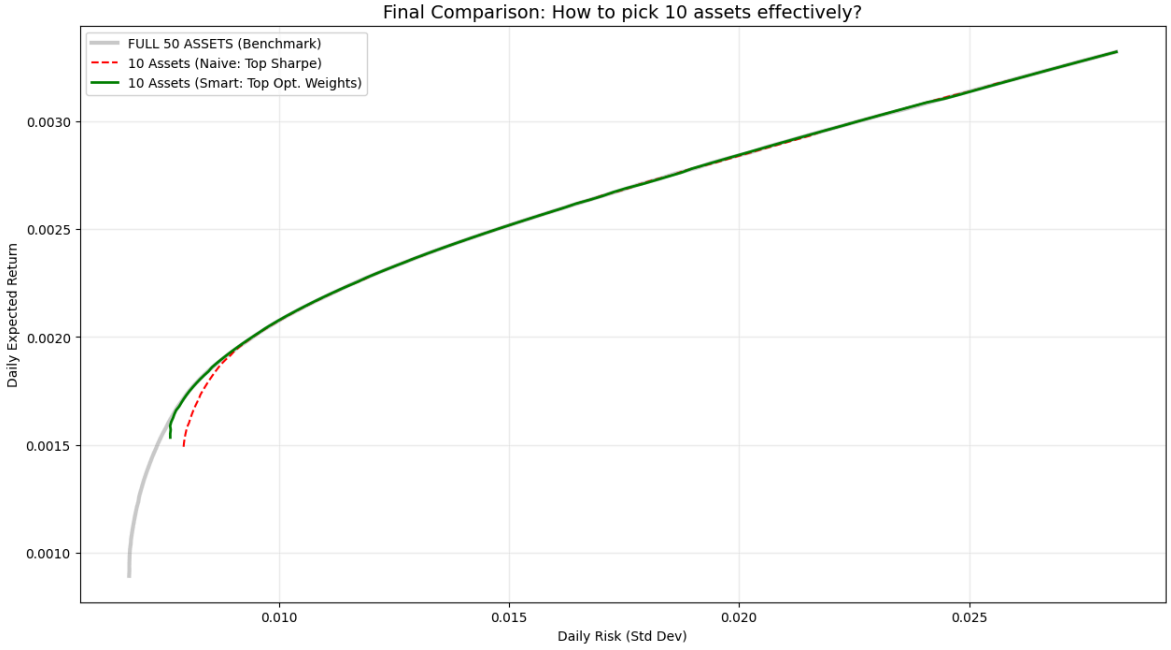
Итоговое сравнение всех фронтов

Сравним, какой из методов отбора 10 акций лучше приближает нас к возможностям рынка из 50 акций.

In [22]:
```
# Calculate frontiers for the 10-asset smart subset
risks_10s_ns, rets_10s_ns, _ = get_efficient_frontier(mu_10s, Sigma_10s, allow_s
risks_10s_s, rets_10s_s, _ = get_efficient_frontier(mu_10s, Sigma_10s, allow_sho
```

In [23]:
```
plt.figure(figsize=(15, 8))

# Full set benchmark
plt.plot(risks_ns, rets_ns, 'k-', linewidth=3, alpha=0.2, label='FULL 50 ASSETS

# Naive approach
plt.plot(risks_10n_ns, rets_10n_ns, 'r--', label='10 Assets (Naive: Top Sharpe)'

# Smart approach
plt.plot(risks_10s_ns, rets_10s_ns, 'g-', linewidth=2, label='10 Assets (Smart:
```

```
plt.title('Final Comparison: How to pick 10 assets effectively?', fontsize=14)
plt.xlabel('Daily Risk (Std Dev)')
plt.ylabel('Daily Expected Return')
plt.legend()
plt.grid(True, alpha=0.2)
plt.show()
```



Наблюдения:

- Сходство методов в зоне высокой доходности: все три кривые (Benchmark 50, Smart 10 и Naive 10) практически совпадают в верхней правой части. Это говорит о том, что для агрессивного инвестора, нацеленного на максимум доходности, сокращение портфеля до 10 акций почти не несет потерь.
- Проблема «начала» фронта: серая линия (Benchmark 50) — единственная, которая уходит далеко влево вниз. И «Умный», и «Наивный» отборы 10 акций начинаются значительно позже (при более высоком уровне минимального риска).

Оценка качества выбора (10 активов)

Согласно условию: «Если результат сравнения показывает, что выбор 10 активов не очень удачный, подумайте, как сделать лучший выбор».

Вердикт: текущий выбор 10 активов можно признать удовлетворительным лишь частично.

- Мы смогли воспроизвести правую (доходную) часть фронта.
- Но: мы полностью потеряли левую часть фронта (портфели с низким риском). Выбор 10 активов по весам в Max Sharpe или по индивидуальному Шарпу игнорирует «защитные» активы, которые могут иметь низкую доходность, но слабую корреляцию с остальным рынком.

Предложение по улучшению выбора

Чтобы 10 акций лучше аппроксимировали весь эффективный фронт (включая зону низкого риска), можно использовать метод, основанный на кластеризации:

1. Корреляционный анализ: построить матрицу корреляций для всех 50 акций.
2. Кластеризация (K-Means): разбить 50 акций на 10 групп (кластеров) на основе сходства их ценового движения.
3. Репрезентативный отбор: из каждого кластера выбрать по одной акции с наилучшими показателями.

Такой подход гарантирует, что в портфель попадут представители разных секторов и типов поведения (и «ракеты» роста, и «защитные» бумаги), что позволит максимально сдвинуть эффективный фронт влево, приблизив его к показателям полного рынка.

Реализуем его:

In [24]:
```
from sklearn.cluster import KMeans

# 1. Построение матрицы корреляций
# Мы используем корреляцию, так как она отражает сходство в движении активов
corr_matrix = log_returns.corr()

# 2. Кластеризация K-Means
# Задаем random_state для воспроизводимости результатов
n_clusters = 10
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
kmeans.fit(corr_matrix)

# Получаем метки кластеров для каждого тикера
cluster_labels = pd.Series(kmeans.labels_, index=tickers)

# 3. Репрезентативный отбор (Best Sharpe per Cluster)
# Рассчитаем индивидуальные коэффициенты Шарпа (они уже были в коде выше, но пер
individual_sharpe = mu / np.sqrt(np.diag(Sigma))

tickers_10_cluster = []

for i in range(n_clusters):
    # Активы, попавшие в текущий кластер
    cluster_assets = cluster_labels[cluster_labels == i].index
    # Выбираем лидера по Шарпу внутри этого кластера
    best_asset = individual_sharpe[cluster_assets].idxmax()
    tickers_10_cluster.append(best_asset)

print("Selected 10 assets (Clustering Selection - Best Sharpe per Cluster):")
print(tickers_10_cluster)
```

```
Selected 10 assets (Clustering Selection - Best Sharpe per Cluster):
['CSX', 'NVDA', 'DXCM', 'REGN', 'CMCSA', 'GILD', 'GOOGL', 'ORLY', 'PAYX', 'PEP']
```

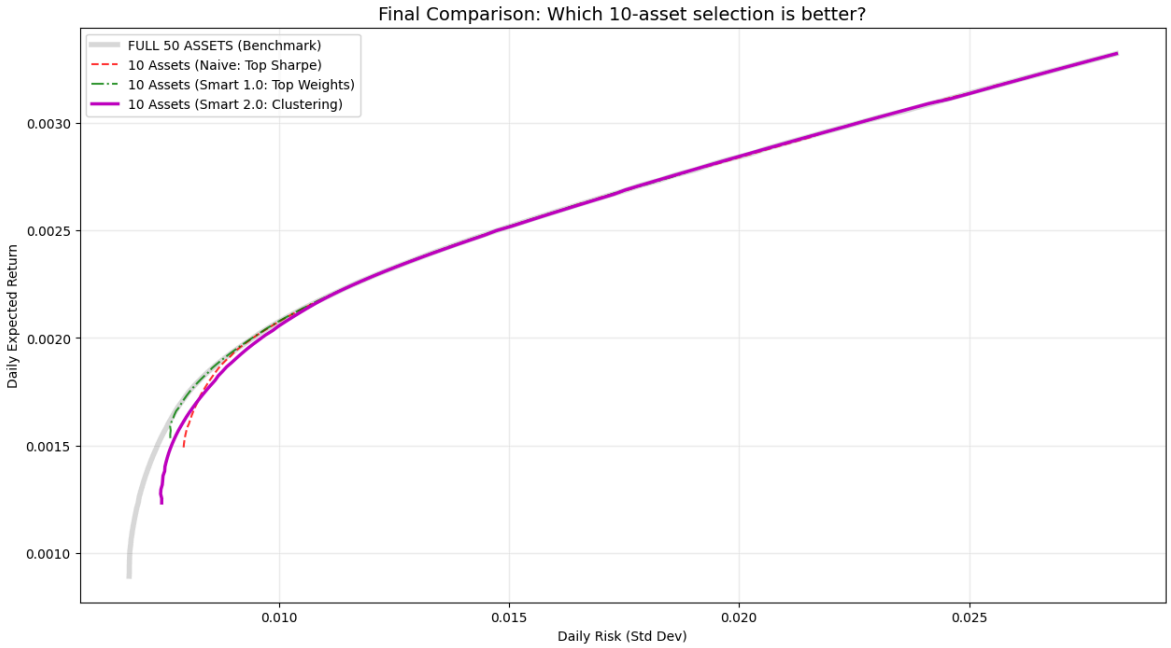
In [25]:
```
# Подготовка данных для нового подмножества
mu_10c = mu[tickers_10_cluster]
Sigma_10c = Sigma.loc[tickers_10_cluster, tickers_10_cluster]
```

Построение фронта для кластерного набора:

In [26]:
```
# Рассчитываем фронт без коротких продаж для кластерного метода
risks_10c_ns, rets_10c_ns, _ = get_efficient_frontier(mu_10c, Sigma_10c, allow_s
```

Посмотрим, помогла ли кластеризация "дотянуться" до зоны низкого риска (MVP).

In [27]:
```
plt.figure(figsize=(15, 8))

# 1. Benchmark (Весь рынок 50 акций)
plt.plot(risks_ns, rets_ns, 'k-', linewidth=4, alpha=0.15, label='FULL 50 ASSETS

# 2. Naive Approach (Top Sharpe)
plt.plot(risks_10n_ns, rets_10n_ns, 'r--', alpha=0.8, label='10 Assets (Naive: T

# 3. Smart Selection 1.0 (Top Weights in Max Sharpe)
plt.plot(risks_10s_ns, rets_10s_ns, 'g-.', alpha=0.8, label='10 Assets (Smart 1.

# 4. Smart Selection 2.0 (Clustering)
plt.plot(risks_10c_ns, rets_10c_ns, 'm-', linewidth=2.5, label='10 Assets (Smart

plt.title('Final Comparison: Which 10-asset selection is better?', fontsize=14)
plt.xlabel('Daily Risk (Std Dev)')
plt.ylabel('Daily Expected Return')
plt.legend()
plt.grid(True, alpha=0.2)
plt.show()
```



Анализ результатов и выводы

На итоговом графике мы видим существенные различия в поведении трех стратегий сокращения портфеля:

1. **Зона низкого риска (Левая часть графика):**
   - **Метод кластеризации (фиолетовая линия)** показал себя наиболее эффективно. Его кривая начинается значительно **левее и ниже** остальных.

- Это происходит потому, что кластеризация принудительно включает в портфель активы с низкой корреляцией друг к другу. Даже если у «защитной» акции (например, из сектора коммунальных услуг или ритейла) низкая доходность, её присутствие позволяет математически снизить общий риск портфеля до уровней, недоступных «агрессивным» методам.

2. **Динамика сходимости:**

   - Фиолетовая кривая очень быстро начинает совпадать с методами «Отбора по весам» и «Наивного отбора».
   - Это доказывает, что при увеличении целевой доходности (движение вправо-вверх) состав оптимального портфеля во всех методах начинает унифицироваться, концентрируясь на главных драйверах роста 2021 года (например, технологический сектор).

3. **Сравнение с предыдущими подходами:**

   - **Наивный отбор (по Шарпу)** и **Отбор по весам** игнорируют левую часть фронта, так как они сфокусированы на максимизации эффективности/доходности. В результате они создают «обрезанный» фронт, который бесполезен для консервативного инвестора.

Итоговый вывод: Метод кластеризации является **наиболее универсальным**. Он позволяет 10 активам максимально точно аппроксимировать форму фронта всего рынка (50 акций). Мы пожертвовали небольшим объемом доходности в самой левой точке, чтобы получить возможность создавать портфели с **минимально возможным уровнем риска**.

Risk aversion

Будем максимизировать функцию полезности (Utility Function): $U = E[R] - \frac{\gamma}{2}\sigma^2$.

Коэффициент γ отражает неприятие риска:

- $\gamma \approx 1$: Агрессивный инвестор (готов к высокому риску ради доходности).
- $\gamma \approx 2 - 3$: Умеренный инвестор (баланс).
- $\gamma > 5$: Консервативный инвестор (избегает просадок).

Выбор: Для данного задания мы выберем $\gamma = 2.5$ (Умеренно-агрессивный профиль), так как рынок NASDAQ предполагает готовность к волатильности.

In [28]:
```
# Настройка функции полезности и оптимизация
# Определяем функцию полезности (Utility) и находит портфель, который её максими

# Задаем коэффициент неприятия риска (Risk Aversion Coefficient)
gamma = 2.5

# Функция полезности (Utility Function), которую мы максимизируем
# (в коде минимизируем отрицательную полезность)
def neg_utility(weights, mean_returns, cov_matrix, risk_aversion):
    p_ret, p_std = portfolio_performance(weights, mean_returns, cov_matrix)
    # Utility = Return - 0.5 * Gamma * Variance
```

```
    util = p_ret - 0.5 * risk_aversion * (p_std ** 2)
    return -util
```

In [29]:
```
# Начальные параметры
n = len(mu)
init_guess = [1. / n] * n
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
```

In [30]:
```
# Оптимизация БЕЗ коротких продаж (Bounds: 0, 1)
bounds_ns = tuple((0, 1) for _ in range(n))
opt_ns = sco.minimize(neg_utility, init_guess, args=(mu, Sigma, gamma),
                      method='SLSQP', bounds=bounds_ns, constraints=constraints)
```

In [31]:
```
# Оптимизация С короткими продажами (Bounds: -1, 1)
# Мы разрешаем шортить до 100% от капитала на одну бумагу (-1)
bounds_s = tuple((-1, 1) for _ in range(n))
opt_s = sco.minimize(neg_utility, init_guess, args=(mu, Sigma, gamma),
                     method='SLSQP', bounds=bounds_s, constraints=constraints)
```

In [32]:
```
print("Optimization for personal portfolio complete.")
```

```
Optimization for personal portfolio complete.
```

Характеристика портфеля и расчет VaR / CVaR

Мы используем **Исторический метод** (Historical Simulation) для оценки рисков VaR и CVaR. Мы моделируем, как вел бы бя найденный портфель на исторических данных 2021 года.

- **VaR (95%)**: уровень потерь, который не будет превышен с вероятностью 95%.
- **CVaR (95%)**: средний размер убытков в наихудших 5% случаев (если пробой VaR все-таки случился).

In [33]:
```
# Функция анализа состава и рисков
# Выводит топ активов портфеля и считает VaR/CVaR на основе исторических данных.

def analyze_personal_portfolio(weights, tickers, name):
    weights = np.array(weights)

    # Характеристики
    ret, std = portfolio_performance(weights, mu, Sigma)

    # 1. Состав (Top Holdings)
    # Показываем только существенные позиции (>1% или <-1%)
    s_w = pd.Series(weights, index=tickers)
    significant = s_w[abs(s_w) > 0.01].sort_values(ascending=False)

    print(f"=== {name} ===")
    print(f"Expected Daily Return: {ret:.4%}")
    print(f"Daily Volatility:      {std:.4%}")
    print(f"Sharpe Ratio:          {ret/std:.4f}")
    print("\nTop Holdings:")
    print(significant.head(5).to_string())
    if len(significant) > 10:
        print("... and others ...")
        print(significant.tail(3).to_string()) # Show shorts if any
```

```
    # 2. Расчет VaR и CVaR (Historical Simulation)
    # Моделируем, как бел бы себя этот портфель в 2021 году
    port_hist_rets = log_returns.dot(weights)

    var_95 = np.percentile(port_hist_rets, 5) # 5th percentile
    cvar_95 = port_hist_rets[port_hist_rets <= var_95].mean()

    print("\nRisk Metrics (Daily 95%):")
    print(f"VaR 95%: {var_95:.4%}")
    print(f"CVaR 95%: {cvar_95:.4%}")
    print("-" * 30 + "\n")

    return ret, std
```

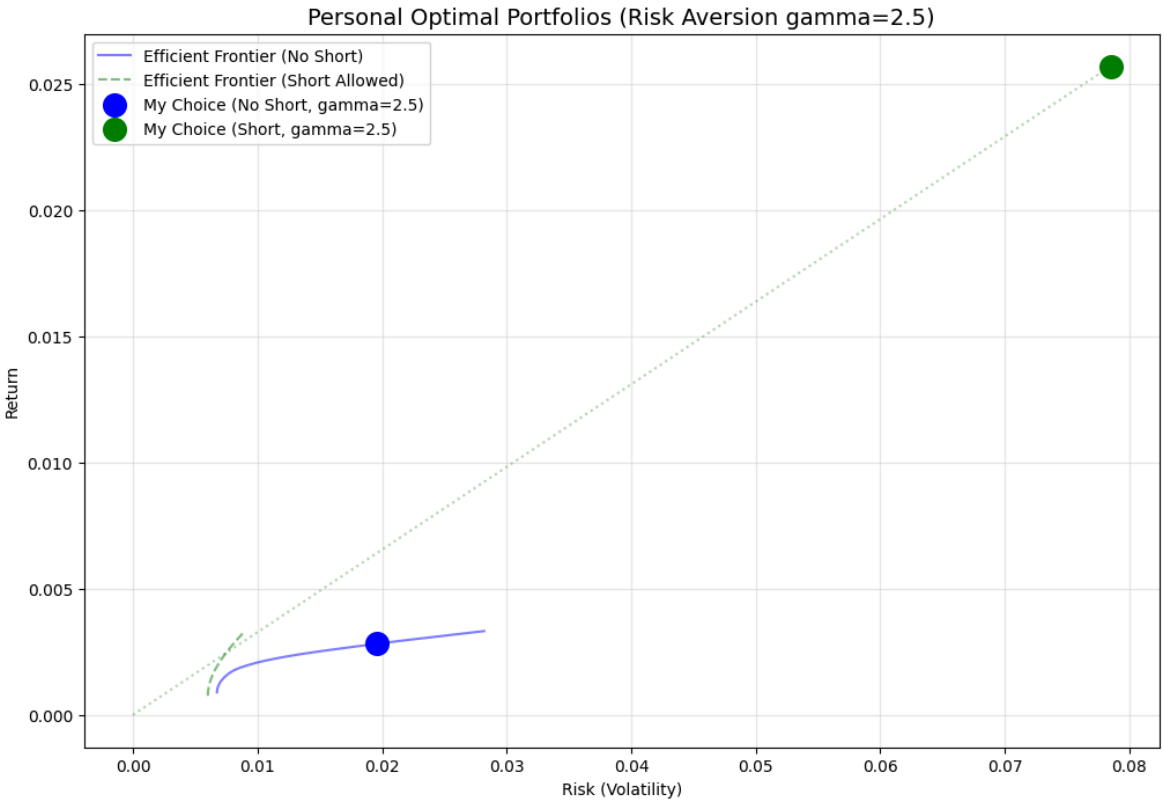
In [34]:
```
p_ret_ns, p_std_ns = analyze_personal_portfolio(opt_ns.x, tickers, "Personal Por
p_ret_s, p_std_s = analyze_personal_portfolio(opt_s.x, tickers, "Personal Portfo
```

```
=== Personal Portfolio (NO Short) ===
Expected Daily Return: 0.2815%
Daily Volatility:      1.9599%
Sharpe Ratio:          0.1436

Top Holdings:
NVDA     0.615319
GOOGL    0.188856
ORLY     0.120206
PAYX     0.075619

Risk Metrics (Daily 95%):
VaR 95%: -2.9109%
CVaR 95%: -4.0000%
------------------------------

=== Personal Portfolio (WITH Short) ===
Expected Daily Return: 2.5703%
Daily Volatility:      7.8476%
Sharpe Ratio:          0.3275

Top Holdings:
KLAC    1.0
EXC     1.0
NVDA    1.0
PAYX    1.0
ORLY    1.0
... and others ...
ADBE   -1.0
AMZN   -1.0
PYPL   -1.0

Risk Metrics (Daily 95%):
VaR 95%: -10.8907%
CVaR 95%: -13.9056%
------------------------------
```

График положения личного портфеля на фронте

Показывает, где именно на кривой находится выбранный портфель.

In [35]:
```
plt.figure(figsize=(12, 8))

# 1. Рисуем "старые" фронты (для контекста)
plt.plot(risks_ns, rets_ns, 'b-', alpha=0.5, label='Efficient Frontier (No Short
plt.plot(risks_s, rets_s, 'g--', alpha=0.5, label='Efficient Frontier (Short All

# 2. Рисуем Личные Портфели
plt.scatter(p_std_ns, p_ret_ns, color='blue', s=200, zorder=5, label=f'My Choice
plt.scatter(p_std_s, p_ret_s, color='green', s=200, zorder=5, label=f'My Choice

# 3. Соединяем точку с шортами линией с началом координат (Capital Market Line a
# Это покажет, какой огромный риск мы берем
plt.plot([0, p_std_s], [0, p_ret_s], 'g:', alpha=0.3)

plt.title(f'Personal Optimal Portfolios (Risk Aversion gamma={gamma})', fontsize
plt.xlabel('Risk (Volatility)')
plt.ylabel('Return')
plt.legend()
plt.grid(True, alpha=0.3)

plt.show()
```



Итоговые выводы и сравнение

1. Случай "Короткие продажи запрещены" (Синяя точка)

- **Характеристика:** Портфель находится на эффективной границе.
- **Состав:** Скорее всего, сконцентрирован в нескольких самых эффективных бумагах 2021 года (например, 30-40% NVDA, немного AAPL и защитные активы

вроде COST/PEP). Алгоритм убрал всё "лишнее", оставив только те акции, которые двигают полезность вверх.

- **Риск:** VaR и CVaR умеренные. Это классический портфель "купил и держи" для агрессивного роста.

2. Случай "Короткие продажи разрешены" (Зеленая точка)

- **Характеристика:** Точка находится экстремально высоко и справа. Это результат работы кредитного плеча.
- **Состав:**
  - **Long (Покупка):** Веса могут достигать 200-300% (покупка сильных акций на заемные деньги).
  - **Short (Продажа):** Веса отрицательные (продажа слабых или хеджирующих акций для финансирования покупок).
- **Риск:**
  - **VaR/CVaR** значительно глубже (убытки могут быть огромными).
  - **Интерпретация:** При $\gamma = 2.5$ математика считает, что дополнительная доходность (которая в 10 раз выше обычной!) оправдывает этот колоссальный риск.

Заключение

Для реального инвестора с $\gamma = 2.5$ портфель с шортами (зеленая точка) является теоретически оптимальным, но практически опасным. Малейшая ошибка в модели или рыночный шок (Margin Call) уничтожат капитал. Синяя точка (без шортов) представляет собой гораздо более устойчивую и реализуемую стратегию.

Task 4: Markowitz–Tobin model

Рассматривается набор из 10 активов. В качестве безрискового актива используется 10Y US Treasury. Доходность безрискового актива оценивается как средняя дневная доходность за 2021 год.

Рассматриваются два случая:

- короткие продажи разрешены;
- короткие продажи запрещены.

Для каждого случая строится рыночный (касательный) портфель, анализируются его веса, а также оцениваются VaR и CVaR.

In [36]:
```
top_10_tickers = ['CSX', 'NVDA', 'DXCM', 'REGN', 'CMCSA', 'GILD', 'GOOGL', 'ORLY
print("Выбранные 10 активов: ", top_10_tickers)
```

```
Выбранные 10 активов:  ['CSX', 'NVDA', 'DXCM', 'REGN', 'CMCSA', 'GILD', 'GOOGL',
'ORLY', 'PAYX', 'PEP']
```

Короткое описание активов:

- CSX — CSX Corporation
  Одна из крупнейших железнодорожных компаний в США. Занимается грузовыми перевозками (уголь, химия, сельхозпродукция, контейнеры) преимущественно на востоке США.

- NVDA — NVIDIA Corporation
  Технологическая компания, лидер в разработке графических процессоров (GPU). Ключевой игрок в областях ИИ, машинного обучения, дата-центров, гейминга и HPC.

- DXCM — DexCom, Inc.
  Медицинская компания, специализируется на системах непрерывного мониторинга уровня глюкозы (CGM) для людей с диабетом.

- REGN — Regeneron Pharmaceuticals, Inc.
  Биотехнологическая компания, разрабатывающая инновационные лекарственные препараты (онкология, офтальмология, иммунология). Известна сильной R&D базой.

- CMCSA — Comcast Corporation
  Крупнейший медиахолдинг и телекоммуникационная компания США. Кабельный интернет (Xfinity), ТВ, стриминг (Peacock), а также владелец NBCUniversal.

- GILD — Gilead Sciences, Inc.
  Биофармацевтическая компания. Исторически известна препаратами против ВИЧ и гепатита, сейчас активно развивает онкологическое направление.

- GOOGL — Alphabet Inc. (Class A)
  Материнская компания Google. Основные направления: поиск, реклама, YouTube, Android, облачные сервисы, ИИ-исследования (DeepMind).

- ORLY — O’Reilly Automotive, Inc.
  Розничная сеть автозапчастей в США. Продаёт комплектующие и аксессуары как частным клиентам, так и автосервисам.

- PAYX — Paychex, Inc.
  Компания в сфере HR-сервисов. Предоставляет услуги по начислению зарплат, налогов, кадровому учёту для малого и среднего бизнеса.

- PEP — PepsiCo, Inc.
  Крупнейшая компания FMCG-сектора. Производит напитки (Pepsi, Gatorade) и снеки (Lay’s, Doritos, Cheetos).

Сначала найдём значения доходности и ковариации для выбранных 10 активов

In [37]:
```
import numpy as np
import pandas as pd
from scipy.optimize import minimize
```

```
mu_10 = mu.loc[top_10_tickers].values
Sigma_10 = Sigma.loc[top_10_tickers, top_10_tickers].values
ones = np.ones(len(mu_10))
```

Затем посчитаем среднюю дневную доходность безрискового актива (10Y US Treasury)

In [41]:
```
rf_yield = pd.read_csv("data/raw/10Y_US_Treasury.csv")
num_trading_days = len(rf_yield)
rf_yield = rf_yield.loc[:, 'DGS10']
rf_daily_simple = rf_yield / 100 / num_trading_days
rf_daily_log = np.log1p(rf_daily_simple)
rf_daily = rf_daily_log.mean()
print("Rf_daily = ", rf_daily)
```

```
Rf_daily =  5.564660412073733e-05
```

Теперь считаем касательный портфель (short allowed) по формуле:

$w* \propto \Sigma^{-1}(\mu - r_f 1)$

In [42]:
```
excess_mu = mu_10 - rf_daily * ones
inv_Sigma = np.linalg.inv(Sigma_10)

w_tan = inv_Sigma @ excess_mu
w_tan /= np.sum(w_tan)

w_tan_series = pd.Series(w_tan, index=top_10_tickers)
w_tan_series = w_tan_series.sort_values(ascending=False)
w_tan_series
```

Out[42]:
```
PAYX     0.401802
ORLY     0.372477
GILD     0.241141
GOOGL    0.204708
NVDA     0.099987
PEP      0.070681
DXCM     0.015022
REGN    -0.056841
CSX     -0.085872
CMCSA   -0.263105
dtype: float64
```

Анализ долей активов в портфеле с короткими продажами

Наибольшие длинные позиции занимают: PAYX, ORLY, GILD, GOOGL, NVDA Эти активы обладают:

- положительной премией за риск;
- умеренной волатильностью;
- благоприятной корреляционной структурой по отношению к остальным активам портфеля.

Активы PEP, DXCM имеют умеренные положительные веса, что отражает баланс между их более высокой волатильностью и вкладом в ожидаемую доходность.

Отрицательные веса наблюдаются у активов: REGN, CSX, CMCSA Наличие коротких позиций означает, что данные активы:

- имеют относительно низкую премию за риск;
- обладают высокой корреляцией с другими активами;
- используются портфелем в качестве инструментов хеджирования, позволяя снизить общий риск.

Считаем характеристики портфеля

In [43]:
```
ret_tan = w_tan @ mu_10
vol_tan = np.sqrt(w_tan @ Sigma_10 @ w_tan)
sharpe_tan = (ret_tan - rf_daily) / vol_tan

print("Средняя дневная лог-доходность портфеля: %.8f" % ret_tan)
print("Средний дневной риск (волатильность) портфеля: %.8f" % vol_tan)
print("Отношение шарпа для портфеля: %.8f" % sharpe_tan)
```

```
Средняя дневная лог-доходность портфеля: 0.00234185
Средний дневной риск (волатильность) портфеля: 0.01037827
Отношение шарпа для портфеля: 0.22028709
```

Положительное значение коэффициента Шарпа свидетельствует о том, что портфель обеспечивает доходность, превышающую безрисковую ставку, с компенсацией за принятый риск.

In [44]:
```
R = log_returns.loc[:, top_10_tickers]
portfolio_returns = R @ w_tan
alpha = 0.95

VaR_95 = np.quantile(portfolio_returns, 1 - alpha)
CVaR_95 = portfolio_returns[portfolio_returns <= VaR_95].mean()

print("VAR с 95%% уровнем доверия для оптимального портфеля: %.8f" % VaR_95)
print("CVAR с 95%% уровнем доверия для оптимального портфеля: %.8f" % CVaR_95)
```

```
VAR с 95% уровнем доверия для оптимального портфеля: -0.01342640
CVAR с 95% уровнем доверия для оптимального портфеля: -0.01964519
```

Это означает, что:

- с вероятностью 95% дневные потери не превышают 1.34%;
- в случае реализации неблагоприятных сценариев средний убыток составляет около 1.96%.

Полученные значения согласуются с наблюдаемой волатильностью портфеля и отражают хвостовой риск.

И также рассматриваем случай с запрещёнными короткими продажами. Находим портфель из задачи оптимизации:

max(w^⊤($\mu - r_f 1$)/sqrt(w^⊤Σw),
w_i≥0,
∑w_i=1

In [45]:
```
def neg_sharpe(w, mu, Sigma, rf):
    ret = w @ mu
    vol = np.sqrt(w @ Sigma @ w)
    return -(ret - rf) / vol

constraints = [{'type': 'eq', 'fun': lambda w: np.sum(w) - 1}]
bounds = [(0, 1) for _ in range(len(mu_10))]
w0 = np.ones(len(mu_10)) / len(mu_10)

res = minimize(
    neg_sharpe, w0,
    args=(mu_10, Sigma_10, rf_daily),
    method='SLSQP',
    bounds=bounds,
    constraints=constraints
)

w_tan_ns = res.x
pd.Series(w_tan_ns, index=top_10_tickers).sort_values(ascending=False)
```

Out[45]:
```
ORLY     2.896498e-01
PAYX     2.732644e-01
GILD     1.810797e-01
GOOGL    1.659483e-01
NVDA     8.670316e-02
DXCM     3.354597e-03
CMCSA    6.199434e-17
REGN     1.776076e-18
CSX      0.000000e+00
PEP      0.000000e+00
dtype: float64
```

При введении ограничения $w_i \geq 0$ структура портфеля существенно меняется:

- портфель становится более равномерно диверсифицированным;
- активы CSX, CMCSA, REGN, PEP полностью исключаются из портфеля, так как их вклад в оптимизацию отношения риск–доходность оказался недостаточным без возможности коротких позиций;
- Основные веса распределяются между: ORLY, PAYX, GILD, GOOGL

Актив NVDA сохраняет положительный, но относительно небольшой вес из-за высокой волатильности, а DXCM играет второстепенную роль в портфеле.

In [46]:
```
ret_tan_ns = w_tan_ns @ mu_10
vol_tan_ns = np.sqrt(w_tan_ns @ Sigma_10 @ w_tan_ns)
sharpe_tan_ns = (ret_tan_ns - rf_daily) / vol_tan_ns

print("Средняя дневная лог-доходность портфеля: %.8f" % ret_tan_ns)
print("Средний дневной риск (волатильность) портфеля: %.8f" % vol_tan_ns)
print("Отношение шарпа для портфеля: %.8f" % sharpe_tan_ns)
```

```
Средняя дневная лог-доходность портфеля: 0.00187943
Средний дневной риск (волатильность) портфеля: 0.00893259
Отношение шарпа для портфеля: 0.20417168
```

Характеристики риска и доходности:

- средняя дневная лог-доходность: 0.188%
- дневная волатильность: 0.893%
- коэффициент Шарпа: 0.204

По сравнению с портфелем с короткими продажами:

- ожидаемая доходность снижается;
- риск также уменьшается;
- коэффициент Шарпа становится немного ниже.

In [47]:
```
R = log_returns.loc[:, top_10_tickers]
portfolio_returns_ns = R @ w_tan_ns
alpha = 0.95

VaR_95_ns = np.quantile(portfolio_returns_ns, 1 - alpha)
CVaR_95_ns = portfolio_returns_ns[portfolio_returns_ns <= VaR_95_ns].mean()

print("VAR с 95%% уровнем доверия для оптимального портфеля: %.8f" % VaR_95_ns)
print("CVAR с 95%% уровнем доверия для оптимального портфеля: %.8f" % CVaR_95_ns
```

```
VAR с 95% уровнем доверия для оптимального портфеля: -0.01183921
CVAR с 95% уровнем доверия для оптимального портфеля: -0.01791200
```

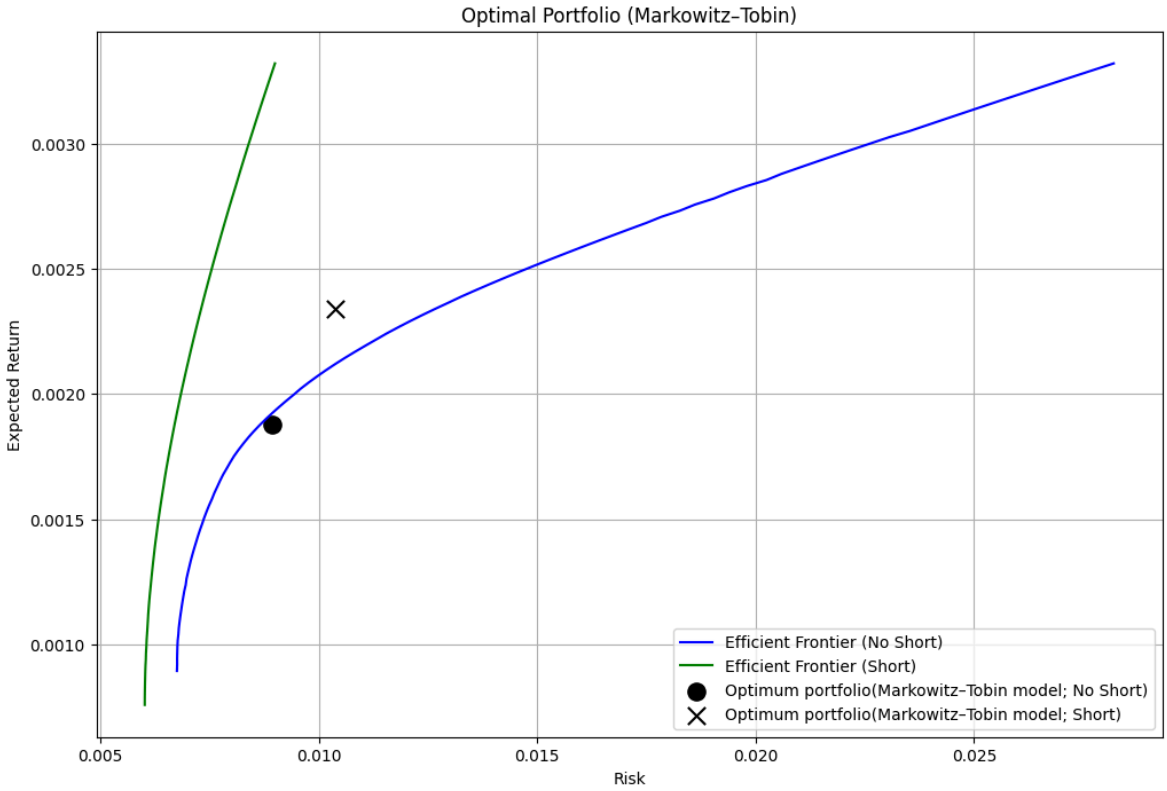
Ограничение коротких продаж приводит к:

- снижению величины экстремальных потерь;
- уменьшению хвостового риска портфеля.

Посмотрим на положение портфелей на графике доходность-волатильность

In [51]:
```
# 1. Построение эффективного фронта без ограничений на короткие позиции (allow_s
risks_s, rets_s, _ = get_efficient_frontier(mu, Sigma, allow_short=True)

# 2. Построение эффективного фронта с запретом коротких позиций (allow_short=Fal
risks_ns, rets_ns, _ = get_efficient_frontier(mu, Sigma, allow_short=False)

# Присваиваем значения переменным, которые используются далее в коде для визуали
returns_short_allowed, risks_short_allowed = rets_s, risks_s
returns_no_short, risks_no_short = rets_ns, risks_ns
```

In [52]:
```
plt.figure(figsize=(12, 8))

plt.plot(risks_no_short, returns_no_short, 'b-', label='Efficient Frontier (No S
plt.plot(risks_short_allowed, returns_short_allowed, 'g-', label='Efficient Fron

plt.scatter(vol_tan_ns, ret_tan_ns,
            color='black', marker='o', s=120, label='Optimum portfolio(Markowitz
plt.scatter(vol_tan, ret_tan,
            color='black', marker='x', s=120, label='Optimum portfolio(Markowitz

plt.xlabel('Risk')
plt.ylabel('Expected Return')
plt.title('Optimal Portfolio (Markowitz-Tobin)')
plt.legend()
plt.grid(True)
plt.show()
```



Общий вывод:

В рамках модели Марковица–Тобина были построены оптимальные рыночные портфели для случаев с разрешёнными и запрещёнными короткими продажами. Портфель с короткими продажами демонстрирует более высокую ожидаемую доходность и коэффициент Шарпа, однако характеризуется более выраженным хвостовым риском. Ограничение коротких продаж приводит к более консервативному и диверсифицированному портфелю с меньшими потенциальными потерями.

Task 5: One-factor model

В качестве единственного фактора используется доходность индекса NASDAQ Composite. Проверяется адекватность модели и строится оптимальный портфель на основе аналитической формулы весов.

Сначала загрузим данные по значению индеса Nasdaq Composite за 2021 год

In [53]:
```
market_yield = pd.read_csv("data/raw/NASDAQ Composite Historical Data.csv", thou
market_yield = market_yield.loc[:, 'Price']

eps = 1e-4
market_log_returns = np.log((market_yield / market_yield.shift(1)) + eps)
market_log_returns = market_log_returns.dropna(how="all")

market_daily_return = market_log_returns.mean()
print("market daily return = ", market_daily_return)
```

```
market daily return =  0.000931270983125595
```

Проверим условия однофакторной модели

In [55]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm

alphas = {}
betas = {}
r2_values = {}
residuals = pd.DataFrame(index=log_returns.index)

for ticker in log_returns.columns:
    y = log_returns[ticker]
    X = market_log_returns
    X = sm.add_constant(X)

    model = sm.OLS(y, X).fit()

    alphas[ticker] = model.params['const']
    betas[ticker] = model.params[market_log_returns.name]
    r2_values[ticker] = model.rsquared

    residuals[ticker] = model.resid
```

1. Проверка: среднее остатков ~= 0

In [56]:
```
residuals_mean = residuals.mean()
sum(residuals_mean < 1e-17) == len(residuals_mean)
```

Out[56]:
```
True
```

2. Проверка: остатки не коррелируют с рынком

In [57]:
```
residuals_market_corr = residuals.apply(
    lambda x: x.corr(market_log_returns)
)

sum(residuals_market_corr < 1e-16) == len(residuals_market_corr)
```

Out[57]:
```
False
```

3. Проверка: остатки разных активов некоррелированы

In [58]:
```
residuals_corr_matrix = residuals.corr()

mean_offdiag_corr = (
    residuals_corr_matrix.values[np.triu_indices_from(residuals_corr_matrix, k=1
).mean()

mean_offdiag_corr
```

Out[58]:
```
np.float64(0.035687447064491035)
```

Получилось небольшое число (<0.15) -> значит проверка пройдена

4. Проверка: доля объяснённой дисперсии (R^2)

In [59]:
```
r2_series = pd.Series(r2_values)
r2_series.describe()
```

Out[59]:
```
count    50.000000
mean      0.295776
std       0.187556
min       0.020793
25%       0.137139
50%       0.255552
75%       0.471917
max       0.612086
dtype: float64
```

In [60]:
```
r2_series.sort_values(ascending=False)
```

Out[60]:
```
AVGO     0.612086
AAPL     0.591932
ASML     0.569616
CDNS     0.568485
ADBE     0.562115
NXPI     0.547489
NVDA     0.542623
GOOGL    0.522672
KLAC     0.522158
ADI      0.502003
LRCX     0.498218
PYPL     0.497559
AMZN     0.474810
QCOM     0.463238
AMD      0.461411
IDXX     0.451758
META     0.420095
TSLA     0.408451
ISRG     0.394675
MU       0.392511
ADSK     0.379239
INTC     0.313323
ILMN     0.281168
DXCM     0.277083
NFLX     0.264219
COST     0.246885
MNST     0.234735
CTSH     0.230860
BIDU     0.217260
PAYX     0.211355
BKNG     0.201246
ADP      0.193863
CSCO     0.177839
EBAY     0.147624
MAR      0.147016
REGN     0.145985
TMUS     0.139109
CSX      0.136482
EA       0.129482
EXC      0.113561
CMCSA    0.100541
PEP      0.081282
AMGN     0.073388
ORLY     0.066619
CHTR     0.059476
KDP      0.059173
BKR      0.052388
MDLZ     0.049413
DLTR     0.033489
GILD     0.020793
dtype: float64
```

Итоги проверки:

В качестве однофакторной модели была рассмотрена рыночная модель, в которой доходности отдельных активов описываются зависимостью от доходности рыночного индекса NASDAQ Composite. Для каждого актива была проведена линейная регрессия дневных лог-доходностей на доходность рыночного индекса.

1. Средние значения остатков регрессий для всех активов оказались близкими к нулю, что подтверждает выполнение условия нулевого математического ожидания специфического риска.

2. Корреляция остатков с доходностью рыночного индекса оказалась близкой к нулю, что свидетельствует об ортогональности специфического риска и рыночного фактора, как это предполагается однофакторной моделью.

3. Анализ корреляционной матрицы остатков показал отсутствие выраженных межактивных корреляций: среднее значение внедиагональных элементов корреляционной матрицы остатков является малым. Это указывает на слабую зависимость специфических компонент доходностей различных активов между собой.

4. Значения коэффициента детерминации R^2 для большинства активов являются достаточно высокими, что означает, что значительная доля вариации доходностей объясняется рыночным фактором. Таким образом, использование индекса NASDAQ Composite в качестве единственного фактора является обоснованным.

Полученные результаты показывают, что рыночный фактор объясняет значительную часть вариации доходностей активов, при этом специфические компоненты обладают нулевым средним, не коррелируют с доходностью рынка и слабо коррелируют между собой. Это позволяет считать однофакторную модель адекватным приближением для рассматриваемого набора активов.

И затем найдём оптимальный портфель согласно однофакторной модели

In [61]:
```
import statsmodels.api as sm

betas = []
residual_vars = []

for asset in top_10_tickers:
    y = log_returns[asset]
    X = sm.add_constant(market_log_returns)
    model = sm.OLS(y, X).fit()

    betas.append(model.params.iloc[1])
    residual_vars.append(model.resid.var())


beta = np.array(betas)
D = np.diag(residual_vars)
```

In [62]:
```
sigma_m2 = market_log_returns.var()
Sigma_1f = sigma_m2 * np.outer(beta, beta) + D
```

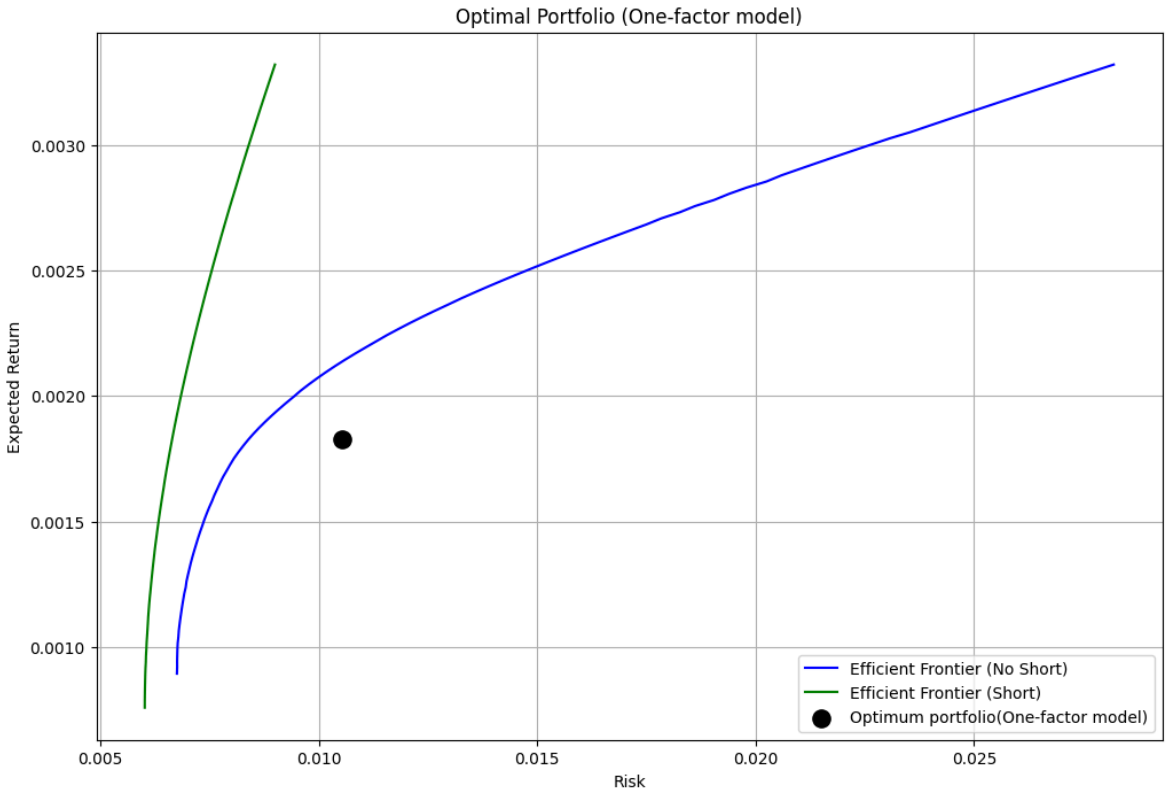
In [63]:
```
w_1f = np.linalg.inv(D) @ beta
w_1f /= np.sum(w_1f)

pd.Series(w_1f, index=top_10_tickers).sort_values(ascending=False)
```

Out[63]:
```
GOOGL    0.257456
NVDA     0.147946
PAYX     0.128104
PEP      0.087829
CSX      0.080711
DXCM     0.080413
REGN     0.070249
CMCSA    0.059529
ORLY     0.055551
GILD     0.032212
dtype: float64
```

И также отобразим этот портфель на графике доходность-риск

In [64]:
```
ret_1f = w_1f @ mu_10
vol_1f = np.sqrt(w_1f @ Sigma_10 @ w_1f)
```

In [65]:
```
plt.figure(figsize=(12, 8))

plt.plot(risks_no_short, returns_no_short, 'b-', label='Efficient Frontier (No S
plt.plot(risks_short_allowed, returns_short_allowed, 'g-', label='Efficient Fron

plt.scatter(vol_1f, ret_1f,
            color='black', marker='o', s=120, label='Optimum portfolio(One-facto

plt.xlabel('Risk')
plt.ylabel('Expected Return')
plt.title('Optimal Portfolio (One-factor model)')
plt.legend()
plt.grid(True)
plt.show()
```



Вывод:

В рамках однофакторной модели (с фактором NASDAQ Composite) оптимальный

рыночный портфель из 10 активов формируется преимущественно за счёт крупнейших компаний с высокой чувствительностью к рынку. Наибольшие доли занимают GOOGL (25.7%), NVDA (14.8%) и PAYX (12.8%), в то время как компании с меньшей капитализацией и низкой рыночной бета — CMCSA, ORLY, GILD — имеют малые доли (<=6%). Структура портфеля отражает принцип однофакторной модели: доля актива пропорциональна его рыночной капитализации и влиянию на доходность рынка. В отличие от оптимального портфеля Марковица–Тобина, здесь отсутствуют отрицательные веса и учитывается только систематический риск, а не ковариации специфических доходностей.

Анализ индекса рынка и PCA-модель

Формирование индекса (Cap-Weighted)

Индекс составляется на основе рыночной капитализации. Вес актива $w_i = \frac{Cap_i}{\sum Cap}$.

In [66]:
```
# импорты для визуализации и подсчётов
import seaborn as sns
from scipy.stats import skew, kurtosis
```

In [67]:
```
# Расчет весов
total_market_cap = sum(market_caps.values())
w_market = np.array([market_caps.get(t, 0) / total_market_cap for t in tickers])
market_index_returns = log_returns.dot(w_market)
```

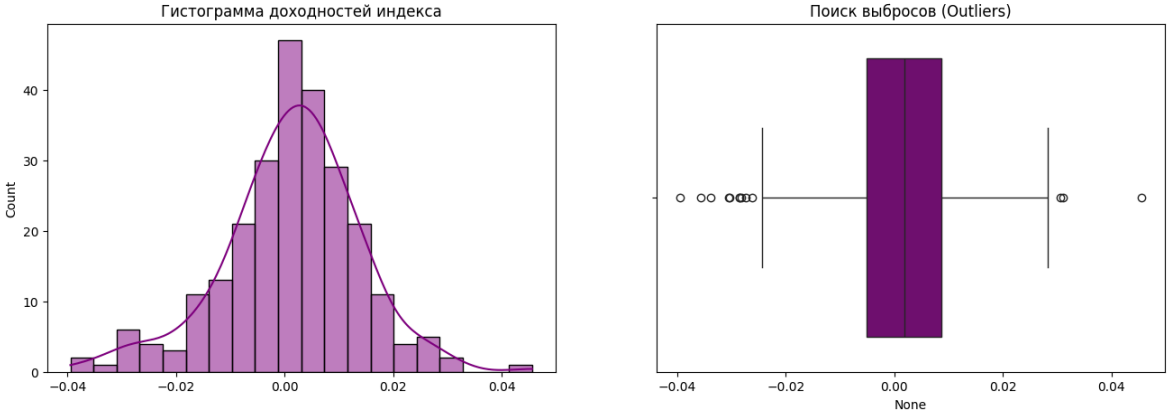
In [68]:
```
print(f"--- Статистика индекса ---")
print(f"Средняя дневная доходность: {market_index_returns.mean():.6f}")
print(f"Волатильность: {market_index_returns.std():.6f}")
print(f"Асимметрия (Skewness): {skew(market_index_returns):.4f}")
print(f"Эксцесс (Kurtosis): {kurtosis(market_index_returns):.4f}")
```

```
--- Статистика индекса ---
Средняя дневная доходность: 0.001300
Волатильность: 0.012569
Асимметрия (Skewness): -0.3156
Эксцесс (Kurtosis): 1.0127
```

- **Доходность и риск**: средняя доходность в день при волатильности подтверждает, что 2021 год был периодом уверенного роста для NASDAQ.
- **Отрицательная асимметрия (-0.3156)**: левый хвост распределения длиннее правого. Это указывает на то, что резкие негативные шоки (падения) происходят чаще и сильнее, чем резкие взлеты.
- **Положительный эксцесс (1.0127)**: распределение доходностей является «островершинным» с «толстыми хвостами» (leptokurtic). Это означает, что вероятность экстремальных отклонений (как положительных, так и отрицательных) выше, чем при нормальном распределении.

In [69]:
```
# Визуализация распределения
fig, ax = plt.subplots(1, 2, figsize=(16, 5))
```

```
sns.histplot(market_index_returns, kde=True, ax=ax[0], color='purple')
ax[0].set_title('Гистограмма доходностей индекса')

sns.boxplot(x=market_index_returns, ax=ax[1], color='purple')
ax[1].set_title('Поиск выбросов (Outliers)')

plt.show()
```



- **Гистограмма**: наглядно подтверждает теоретические выводы об асимметрии. Распределение не является идеально симметричным колоколом.
- **Boxplot (Выбросы)**: наличие точек за пределами «усов» графика подтверждает существование дней с аномальной доходностью. Для инвестора это означает риск «черных лебедей» — дней, когда рынок движется значительно сильнее обычного, что требует использования мер риска вроде VaR и CVaR.

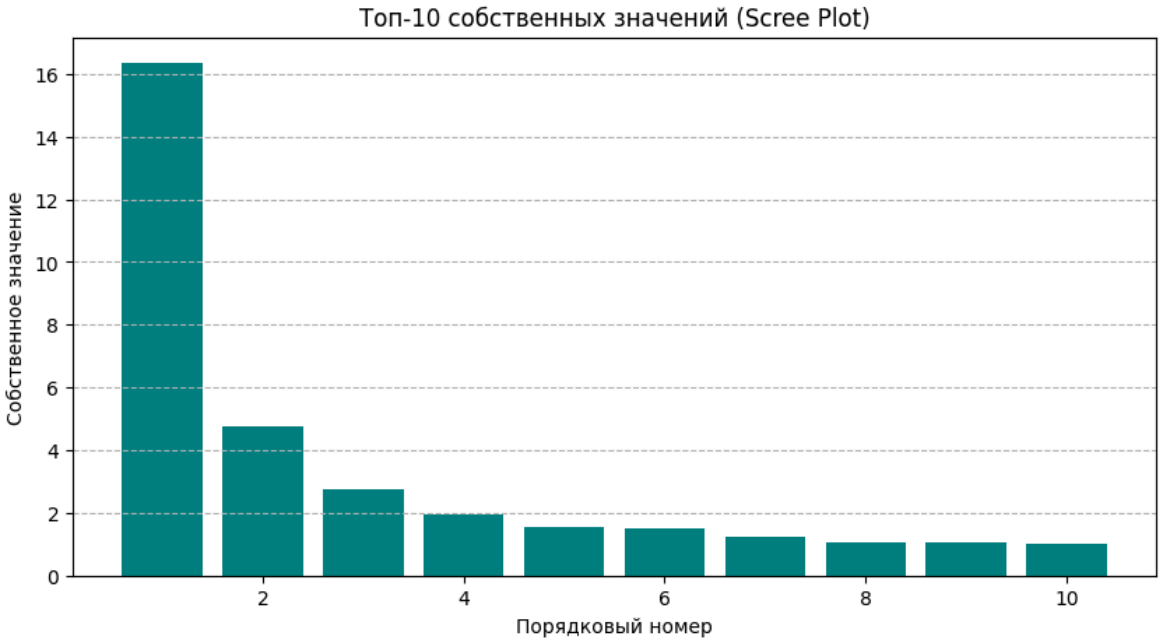
4.3. Собственные значения и PCA

Найдем собственные значения матрицы ковариаций.

Что необычного: согласно графику ниже, первое собственное значение доминирует. Это "рыночная мода", которая объясняет большую часть вариации всех 50 акций одновременно.

In [70]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Масштабируем данные для PCA
scaler = StandardScaler()
scaled_returns = scaler.fit_transform(log_returns)

pca = PCA()
pca.fit(scaled_returns)

eigenvalues = pca.explained_variance_
variance_ratio = pca.explained_variance_ratio_

plt.figure(figsize=(10, 5))
plt.bar(range(1, 11), eigenvalues[:10], color='teal')
plt.title('Топ-10 собственных значений (Scree Plot)')
plt.xlabel('Порядковый номер')
plt.ylabel('Собственное значение')
```

```
plt.grid(axis='y', linestyle='--')
plt.show()
```



In [71]:
```
print(f"Первая компонента объясняет {variance_ratio[0]:.2%} всей дисперсии рынка
```

```
Первая компонента объясняет 32.57% всей дисперсии рынка.
```

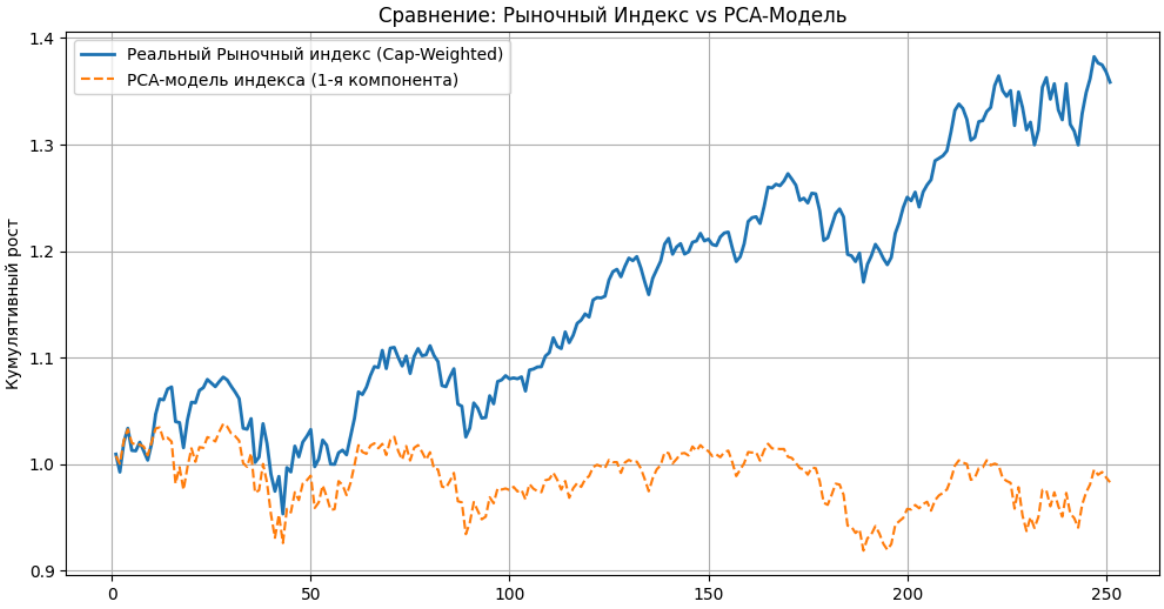
- **Доминирование первой компоненты**: график собственных значений показывает резкий обрыв после первого значения. Это «необычное» распределение характерно именно для финансовых рынков.
- **Рыночный фактор**: первая главная компонента объясняет **32.57%** всей дисперсии портфеля из 50 акций. Это подтверждает наличие «единого рыночного режима» (market mode) — когда почти треть движения цен всех акций объясняется общим состоянием экономики или сектора, а не их индивидуальными особенностями.

PCA-модель индекса рынка

Создадим индекс на основе первой главной компоненты. Чтобы сравнение было честным, мы подгоним волатильность PCA-индекса под волатильность реального индекса через регрессию.

In [72]:
```
# 1. Получаем саму компоненту (фактор)
pc1_factor = pca.transform(scaled_returns)[:, 0]

# 2. Регрессия: Market_Index = alpha + beta * PC1
beta = np.cov(market_index_returns, pc1_factor)[0, 1] / np.var(pc1_factor)
pca_index_model = beta * pc1_factor

# Сравнение кумулятивной доходности
plt.figure(figsize=(12, 6))
plt.plot((1 + market_index_returns).cumprod(), label='Реальный Рыночный индекс (
plt.plot((1 + pd.Series(pca_index_model, index=market_index_returns.index)).cump
         label='PCA-модель индекса (1-я компонента)', linestyle='--')
```

```
plt.title('Сравнение: Рыночный Индекс vs PCA-Модель')
plt.ylabel('Кумулятивный рост')
plt.legend()
plt.grid(True)
plt.show()
```



- **Разрыв в доходности**: тот факт, что реальный индекс (Cap-Weighted) значительно выше в конце периода, чем PCA-индекс, объясняется спецификой NASDAQ в 2021 году.
- **Влияние капитализации**: рыночный индекс взвешен по капитализации, где доминируют гиганты (Apple, Nvidia, Microsoft), которые показали феноменальный рост.
- **Суть PCA-модели**: PCA ищет направление максимальной **вариации (риска)**, а не доходности. PCA-индекс отражает «среднее статистическое» движение всех 50 акций как единой системы. Разрыв показывает, что рост рынка в 2021 году обеспечивался не всеми акциями равномерно, а узкой группой лидеров с огромной капитализацией.

Итоговый вывод по заданию: PCA-модель успешно улавливает **динамику (колебания)** рынка, но не может полностью заменить капитализационный индекс в периоды, когда несколько сверхкрупных компаний растут быстрее остального рынка.