

# ПРАКТИКА



# проект

# Друзья,

Рады приветствовать вас на практике в компании Sibdev!

Летняя практика всегда была возможностью прокачать навыки программирования и с пользой провести лето. Мы пошли дальше — и попытались сделать ее увлекательной, добавив элементы игры.

Задание максимально приближено к тому, с чем вы столкнетесь в будущем на реальных проектах. Приступить к его выполнению можно с минимальным уровнем подготовки в технологии. А вот чтобы успешно закончить, потребуется приложить усилия.

Чтобы научиться профессионально разрабатывать на R, недостаточно лекций и онлайн-курсов. Мы уверены, что реальное обучение — это писать код, ошибаться и пробовать снова. Поэтому предлагаем сделать это вместе на Практике.

Желаем успехов от всей команды Sibdev ❄️

# Задания

В ходе практики мы разработаем фронтенд веб-приложения. Проект в этом году - приложение для учета доходов и расходов, а еще трекинга финансовых целей! 🏆

Вас ждут 11 последовательных этапов разработки. Их сложность будет возрастать, как правило. Поэтому, чтобы лучше разобраться, мы приложили ссылки на документацию и гайды в самих заданиях и в конце документа.

Если не получается самостоятельно решить задание, просите помощи у других практикантов в чате направления и в последнюю очередь — у наставника. Вы также можете помогать коллегам советом.

## Баллы

За задание можно получить от 1 до 10 коинов.

Если вы получили от 1 до 6 коинов за задание — увы, его придется переделать. При этом максимум вы сможете получить 8 баллов.

Если оценка 7, то можно перейти к следующему заданию. А можно повысить оценку до 8, выполнив правки наставника.

Оценку 8–10 повысить нельзя — и так круто. Но можно доработать задание, чтобы утолить перфекционизм.

## Достижения

За определенные действия, свойственные кодеру, вы получите достижения. За каждое достижение — коины. Достижения описаны в [разделе «Достижения»](#).

# Финал практики

Конец практики — 30 июля. Последний день приема и проверки задач - 29 июля.

В [разделе «Маркет»](#) вас ждет стильный мерч. Его можно получить за коины после окончания практики.

По итогам практики мы пригласим лучших практикантов на стажировку.

## Вам понадобятся

- Базовые знания HTML, CSS, JavaScript;
- Node.js актуальной версии ([nodejs.org](https://nodejs.org));
- Yarn актуальной версии ([yarnpkg.com](https://yarnpkg.com)), допустимо использовать npm ([npmjs.com/get-npm](https://npmjs.com/get-npm)).

# Проект

*Попадая в компанию вы почти никогда не будете работать над проектом с нуля. Обычно вы присоединяетесь к уже существующему проекту или шаблону проекта, в котором создана основа или есть “прошлая” версия.*

Представим, что так произошло в этот раз. У нас есть проект Pockets(карманы) версии 1.5, созданный в прошлом году одним из практиков и проапгрейженный нашим коллегой. Таков Lore.

Мы хотим прокачать его и обновить до версии 2.0

Нашем проектом станет приложение для учета средств и получении простой аналитики по категориям расходов и доходов, а также ведения финансовых целей. Приложение будет иметь 3 страницы + регистрацию.

## **Первая страница: Дашборд**

Это стартовая страница с возможностью “быстрого доступа” ко всем ключевым действиям. Если пользователь хочет внести расход или проверить цели - ему сюда.

## **Вторая страница: Аналитика**

Участовавшим в прошлом году она покажется знакомой. Здесь пользователь вносит транзакции, получает аналитику по категориям и видит общее положение своих финансов. Новшеством версии 2.0 станет круговая диаграмма для визуализации данных и возможность выгрузить своих транзакции в виде .xlsx таблицы.

## **Третья страница: Цели**

Здесь мы будем имитировать вклады в банк на цель. Например купить новую видеокарту или накопить на поездку в Черногорию. Деньги на счета целей можно класть под процент и мониторить достижения.

Общий макет: [Pockets 2.0](#)

## Задание 1: Новый дизайн форм входа и регистрации

Как писали выше - нам предстоит работа с исходным проектом. Это прошлая “неклассная” версия. Мы сделаем классную.

Для начала выгрузим себе исходник. Покопаемся в коде, потыкаем кнопки, посмотрим какой функционал здесь есть и что приложение умеет.

Разрабатывать код будем в локальном репозитории, а загружать результаты — в удаленный. Там их будет проверять наставник. Договоримся: при каждой отправке `merge request` будем указывать номер задания и описывать выполненную работу в описании.

Для начала работы с приложением нам нужно создать аккаунт и залогиниться.

Так же нужно будет обновить дизайн форм входа и регистрации:

Старый	Новый
<div><div><h2>Pockets</h2><p>Adventure starts here</p><p>Make your app management easy and fun!</p><p>Username</p><input type="text" value="johndoe@gmail.com"/><p>Email</p><input type="text" value="johndoe@gmail.com"/><p>Password</p><input type="password" value="••••••••"/><p><input type="checkbox"/> Я со всем согласен отпутите</p><p>Sign Up</p><p>Already have an account? <a href="#">Sign in instead</a></p></div></div>	<div><div><h2>Регистрация</h2><p>Сделайте управление вашими финансами простым и увлекательным!</p><p>Введите имя пользователя</p><p>Введите почту</p><p>213123</p><p><input checked="" type="checkbox"/> Я со всем согласен отпутите</p><p>Зарегистрироваться</p><p>У вас уже есть аккаунт? <a href="#">Авторизоваться</a></p></div></div>

## Ход работы

1. В репозитории ответвиться от ветки `master` и создать ветку `feature/#1-update-readme`.
2. Создать несколько новых транзакций и категорий.
3. Добавить в README краткое описание приложения и инструкцию по запуску. Рекомендации по написанию файла можно найти по этой [ссылке](#).
4. Создать вторую ветку от `master` - `feature/#1-update-signup-screen-designs`
5. Обновить дизайн страницы регистрации в соответствии с макетом ([этим макетом](#)). Запустить изменения в соответствующую ветку.
6. Создать два `merge request`'а новых веток в `master`. Отправить ссылки наставнику.

**Мы зарегистрировали аккаунт, разобрались с функционалом приложения, обновили дизайн страницы регистрации и готовы приступить к обновлению.**

# Задание 2: Обновление сервиса запросов

**Мы работали только с одним JWT access без времени жизни. Теперь это будет access/refresh система.**

Нужно найти все места, где производятся запросы на сервер с применением ключей и изменить эти функции. Перевести запросы, использующие axios на встроенный браузерный Fetch API. Если хочется - оформить в ООП стиле\*

Сервер после авторизации будет присылать пару access/refresh токенов, которые будут использоваться для валидации запросов. У access токена будет время жизни. Если попытаться сделать запрос с просроченным токеном - сервер отправит ошибку (HTTP 401). В этом случае нужно будет обновить access token с помощью refresh.

## Ход работы

1. Предыдущие request'ы должны быть слиты в master наставником. После этого создаем новую ветку от master - feature/#2-add-jwt-refresh
2. Перепроектировать функции/классы, через которые происходят запросы на сервер. Они должны учитывать, что токен может протухнуть. Обновляться токен тоже должен прямо в этих функциях
3. Добавить в readme файл пункт, отражающий систему авторизации запросов. Должно быть понятно, что бэкенд теперь защищен двойной системой JWT токенов.
4. Отправить merge request в master наставнику.

**Аутентификация работает корректно, выглядит красиво. Вход приложения - готов.**



## Справка: как работает аутентификация

На странице есть:

- Заголовок и вступительное предложение;
- Поле Логин, Email и Пароль
- Чекбокс (забавный)
- Кнопка “Зарегистрироваться”
- Кнопка для редиректа на страницу “Авторизация”

Клик по кнопке редиректа, что логично, приведет нас на вторую страницу “Авторизация”, эта страница позволит войти в систему с уже созданного аккаунта.

На странице есть:

- Заголовок и вступительное предложение;
- Поля Email и Пароль
- Кнопка “Войти” или “Login”
- Кнопка для редиректа на страницу “Регистрация”

Клик по кнопке редиректа, что логично, приведет нас на страницу “Регистрация”, тем самым закольцевав аутентификацию.

# Задание 3: Страница Операций: Модалки

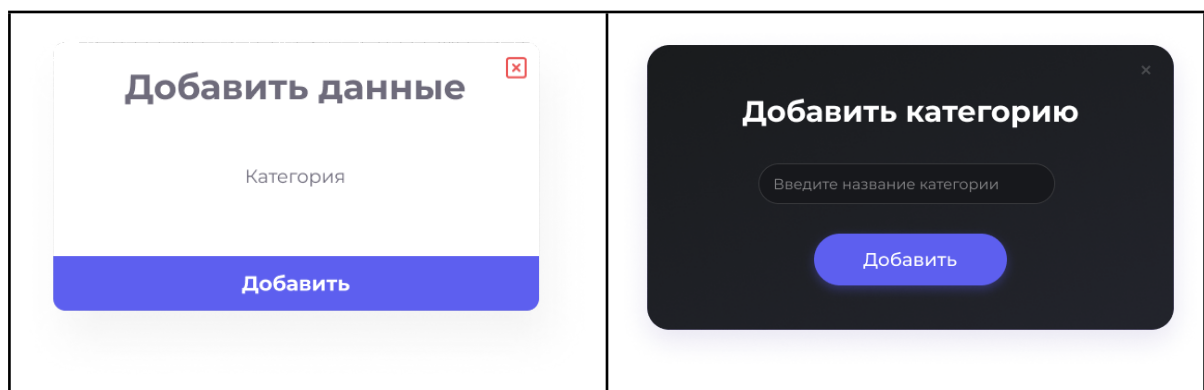
## Переработка дизайна модальных окон страницы Операции.

Мы сделали красивый вход. Самое время сделать, то ради чего пользователь им воспользуется.

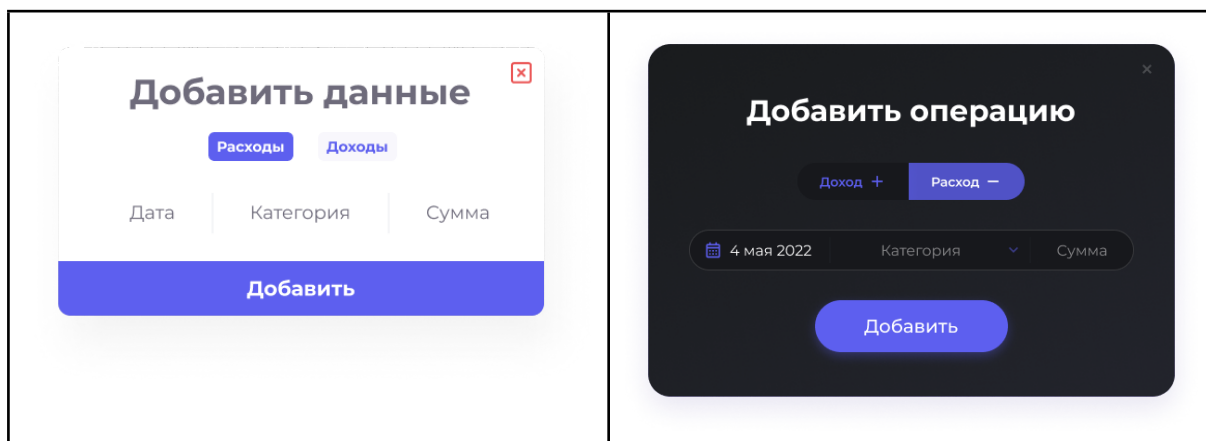
Страница Операции содержит максимум уже готовых компонентов. Но переезд - не простая задача, поэтому начнем с модальных окон.

## Ход работы

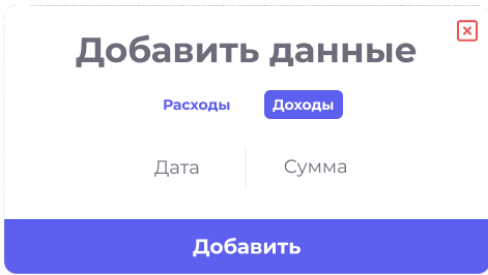
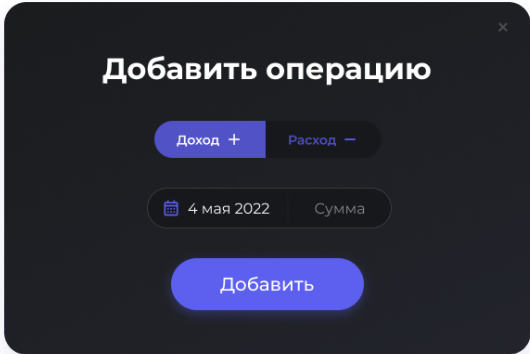
1. Создать ветку `feature/#3-popup-designs`
2. Модалка "Создать категорию". Именные категории будут только у расходов.



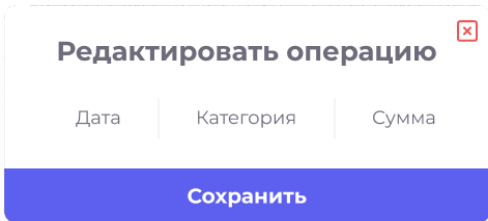
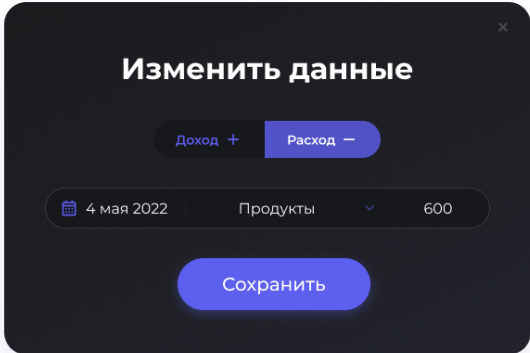
3. Модалка "Создать операцию".
  - а. По клику на дату будет выводиться календарь. Его можно взять готовый из библиотеки, например ([ссылка](#)).



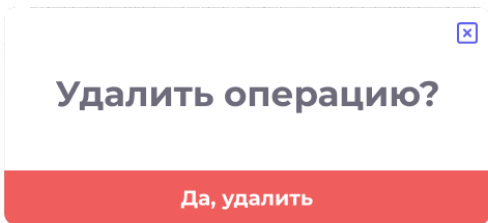
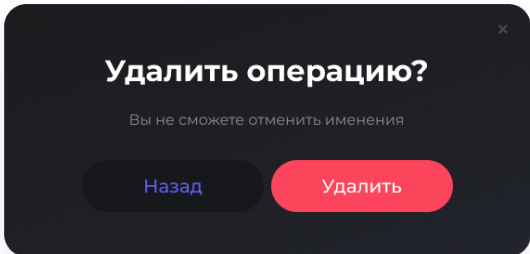
4. Состояние модалки при переключении на “Доход”

 <p>Добавить данные</p> <p>Расходы   <b>Доходы</b></p> <p>Дата   Сумма</p> <p>Добавить</p>	 <p>Добавить операцию</p> <p><b>Доход +</b>   Расход -</p> <p>4 мая 2022   Сумма</p> <p>Добавить</p>
---	--

5. Модалка Редактирование операции

 <p>Редактировать операцию</p> <p>Дата   Категория   Сумма</p> <p>Сохранить</p>	 <p>Изменить данные</p> <p>Доход +   <b>Расход -</b></p> <p>4 мая 2022   Продукты   600</p> <p>Сохранить</p>
---	---

6. Модалка Удаление операции

 <p>Удалить операцию?</p> <p>Да, удалить</p>	 <p>Удалить операцию?</p> <p>Вы не сможете отменить именения</p> <p>Назад   Удалить</p>
---	---

7. Отправить merge request в master наставнику.





# Задание 4: Верстка страницы Операций

**Верстка страницы Операции. Перенос и переработка компонентов с прошлого года. Операции, категории, общее.**

Мы обновили модалки. Приступим к более сложной задаче и обновим дизайн страницы Операции. Нашей задачей станет правильно расположить их на странице и обновить их работу в соответствии с лучшим UX.

Было ([Ссылка](#))

## Карманы

ДАТЫ	КАТЕГОРИЯ	СУММА	Добавить +
01.06.2021	Технологии	10.4k	 
01.06.2021	Технологии	23.4k	 
01.06.2021	Технологии	23.4k	 
01.06.2021	Технологии	12.4k	 
01.06.2021	Технологии	11.4k	 
01.06.2021	Технологии	23.4k	 
01.06.2021	Технологии	23.4k	 

Категории	Добавить +
Расходы	Сумма
Технологии	999999k
Игры	999999k
Счета	999999k
Рестораны	999999k
Продукты	999999k
Дом	999999k
Спорт	999999k
Здоровье	999999k
Транспорт	999999k
Хобби	999999k



Привет, Leo

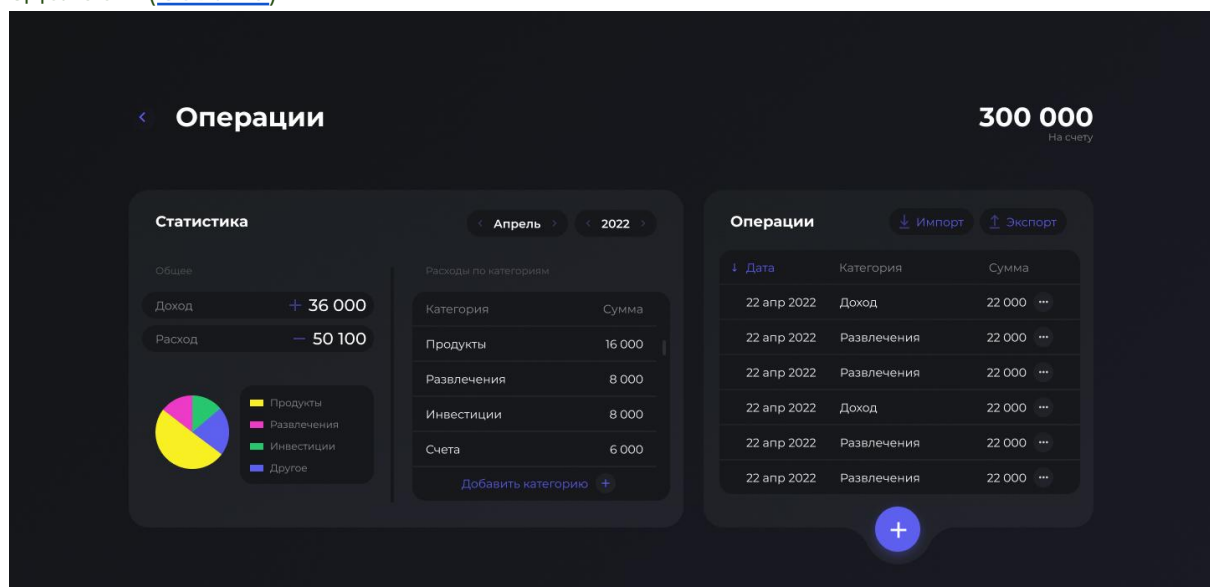
### Февраль 2021

Доход 99999999k

Расход 99999999k

Приколдесное поле

Сделаем ([Ссылка](#))



## Ход работы

1. В репозитории ответить от `master` и создать ветку `feature/#4-operations-page-design`.
2. Разберемся с **таблицей Операций**
  - a. Перенесем кнопку “Добавить операцию” под таблицу.
  - b. Переверстаем модальное окно “Добавить операцию” к новому виду. Добавим заголовок.
  - c. Спрячем действия с операциями для лаконичности
    - i. Заменим 2 иконки - “Редактировать” и “Удалить” многоточием.
    - ii. Создадим выпадающий список. Он будет появляться при клике на многоточие.
    - iii. Его опции будут работать так же как кнопки “Редактировать” и “Удалить” раньше.
  - d. Создадим кнопки “Импорт” и “Экспорт”. Пока клик по ним ничего делать не будет. Семантически это все равно будут `<button>`
3. Переверстаем **таблицу Категории**.
  - a. Перенесем кнопку “Добавить” вниз, под таблицу.
  - b. Добавим заголовок для таблицы - “Расходы по категориям”
4. Обновим блок **Общее**
  - a. Добавим отсечки пробелами чисел от 10 000.
  - b. Сделаем заголовок.
  - c. Выделим место под круговую диаграмму. Пока оставим его пустым.
5. Переместим блоки на новые места. **Общее** - слева, **Категории** - по центру, **Операции** - справа.
6. Создадим “пустое состояние” окон ([Ссылка](#)) Состояние, когда данных нет.
7. Отправим merge request в master наставнику.

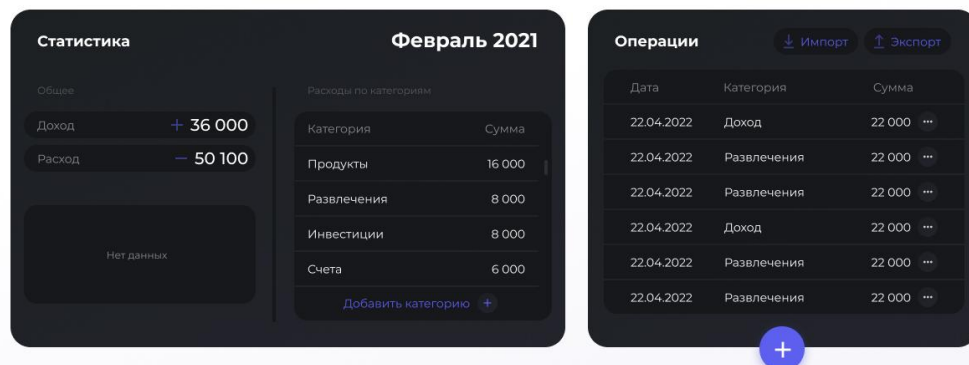


**СУЩЕСТВУЮТ ОСОБЫЕ ОПЕРАЦИИ - ОПЕРАЦИИ С ЦЕЛЯМИ. ОНИ СИСТЕМНЫЕ И ИХ НЕЛЬЗЯ УДАЛИТЬ ИЛИ РЕДАКТИРОВАТЬ. ОНИ ПОЯВЯТСЯ ПОЗЖЕ, ПРОСТО ПРЕДУПРЕЖДАЕМ)**

## Карманы



Привет, Leo



# Задание 5: Счет аккаунта и работа с операциями

**Обновление шапки. Сортировка по столбцам таблицы операций.  
Экспорт и импорт таблицы.**

У нас есть доходы и расходы в месяц. Как насчет понимать, сколько денег у нас всего остается в аккаунте? Создадим **Счет Аккаунта**. Он будет загружаться с сервера на основе текущего месяца/года в окне “Статистика”.

Пора анализировать расходы, проагрейдем **таблицу Операции** фильтрами по сумме, категории и дате. Добавим кнопкам импорта и экспорта **таблицы Операции** соответствующие запросы на сервер. Кнопка “импорт” должна вызывать диалог выбора файла.

## Ход работы:

### Часть 1. Счет Аккаунта

1. В репозитории ответить от `master` и создать ветку `feature/#5-account-balance`.
2. Вместо поля приветствия будем выводить **Счет Аккаунта**. Загружаем данные с сервера. Они представляют счет аккаунта..
3. Вместо названия приложения в шапке поставим название вкладки и добавим ссылку. Она в будущем будет вести на другую страницу - **Дашборд**.

### Часть 2. Сортировка операций

1. Создать новую ветку от `master` - `feature/#5-operations-table-sorting`
2. Сортировка по столбцам таблицы операций. Сортировка должна включать направления вверх и вниз по 3 полям Дата, Сумма, Категория (по алфавиту/по сумме).
  - a. Включение сортировки происходит при клике на шапку столбца. Включается сортировка “по возрастанию”. Второй клик заменяет сортировку в столбце на “по убыванию”. Третий отменяет сортировку.
  - b. Включение сортировки по другому столбцу отключает сортировки по остальным столбцам

### Часть 3. Импорт/экспорт операций

1. Создать новую ветку от `master` - `feature/#5-import-export-operations`
2. Привяжем к кнопке Экспорт запрос на сервер с выгрузкой таблицы операций по активному месяцу.
3. Теперь импорт. Клик по кнопке будет вызывать диалог выбора файла с компьютера пользователя. Выбранный файл будет послан на сервер.

Отправляем все 3 merge request в `master` наставнику.

# Задание 6: Визуализация данных

## **Диаграмма и выбор периода.**

Добавим логику выбору периода на окне “Операции”. Изначально он будет равен текущему месяцу и году. Лучше будет сохранять выбранный период в глобальное состояние приложения. Изменение периода запросит новые данные счета аккаунта и операций.

Данные лучше видны, через цвет и форму, чем через числа. Сделаем диаграмму по логике - 3 самые популярные категории расходов из загруженных и все остальные соберем в “Другое”. Лучше использовать для рендера какую-нибудь библиотеку.

## **Ход работы:**

1. Ответвиться от `master` создать ветку `feature/#6-data-analysis`
2. Добавить возможность выбора периода (месяц/год), за который будет отображаться статистика (Операции на этой странице и диаграмма).
3. Подключить любую библиотеку для построения диаграмм (`chart.js`, `highchart...`) на свой вкус и вывести чарт в блоке, оставленном на окне “Статистика”.
  - a. Чарт будет отображать количество потраченных денег на категории.
  - b. Выведено будет от одной до четырех категорий. Если категорий трат всего 4 - нужно выводить все. Если больше - выводить только три самые затратные, а остальные объединять в один раздел чарта “другие”.
  - c. Наведение на сектор диаграммы должно триггерить подсказку (тултип) с названием категории и ее содержанием в пироге в процентах. Дизайн не принципиален. Можно использовать встроенные средства библиотеки диаграмм.
4. Отправить `merge request` в `master` наставнику.

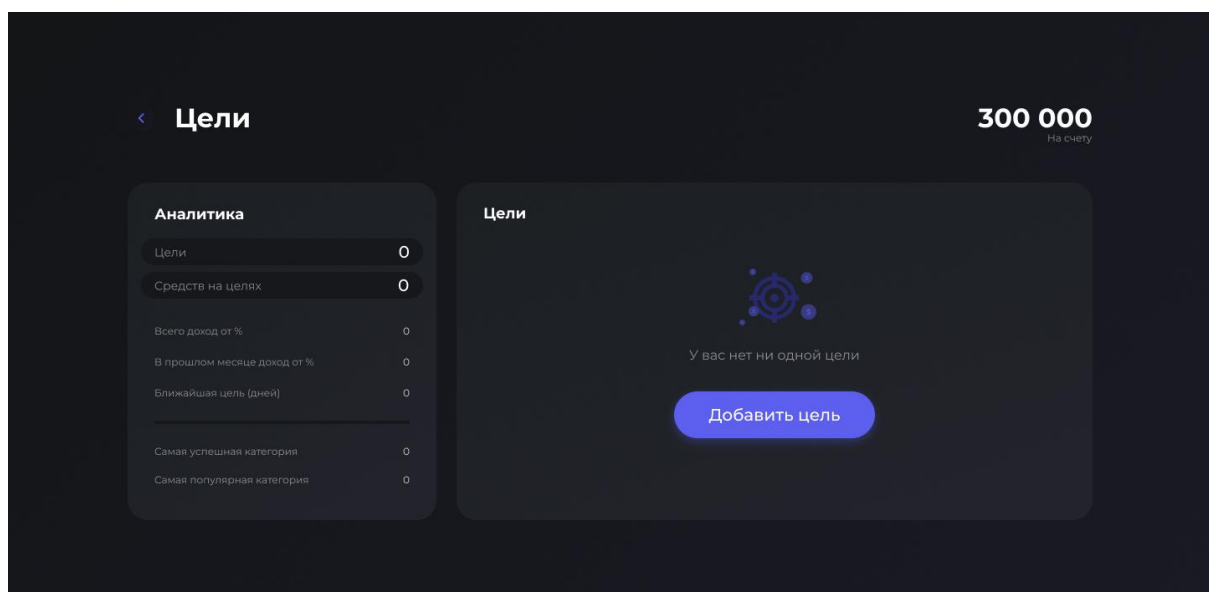
# Задание 7: Цели

**Верстка страницы Целей. Верстка Аналитики. Создание цели.**

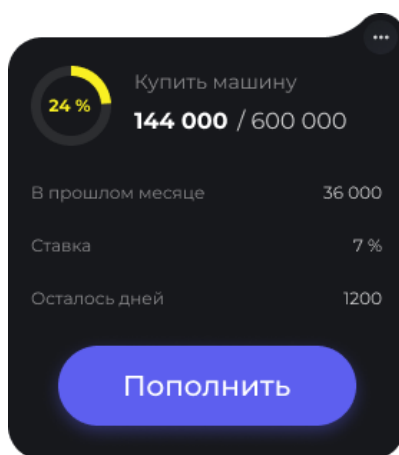
Страница Операции была апгрейдом проекта Практики`21. Как насчет создать что-то принципиально новое?

Цели! Цели позволят юзеру вкладывать часть средств с **Счета Аккаунта** в вклад под процент. А в блоке Аналитика можно будет увидеть общее положение дел по всем целям.

Сверстаем страницу. Значения всех полей будут пустыми.

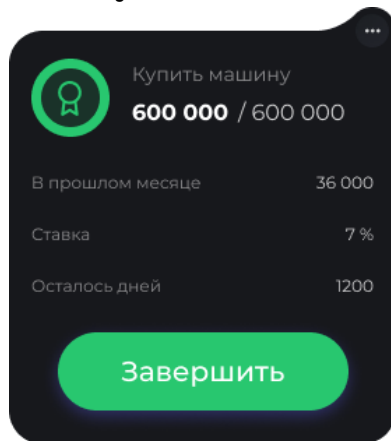


Сверстаем карточку Цель

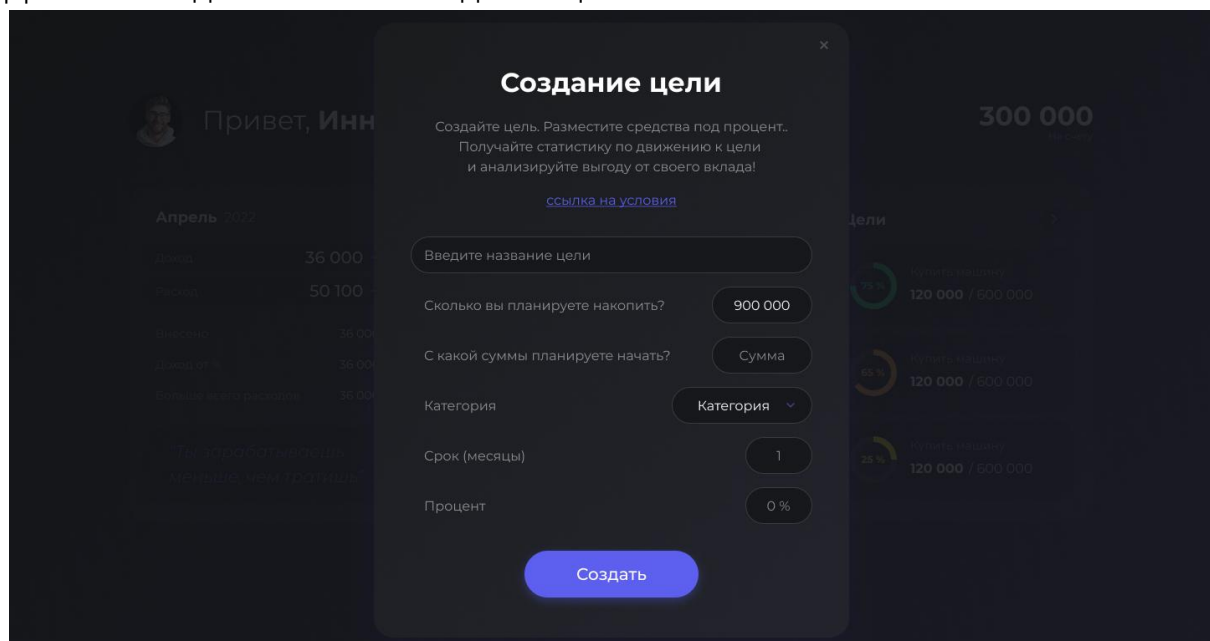




Когда сумма на целе = или больше цели, модалка обновляется к такому виду:



Добавим модальное окно создания цели

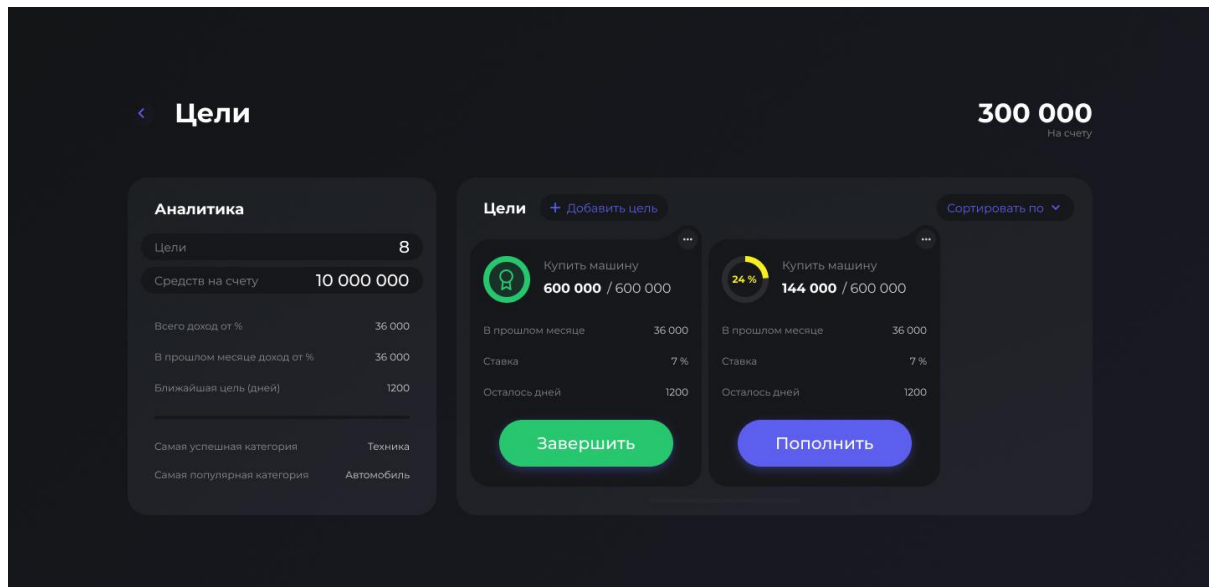


## Ход работы:

1. В репозитории отойти от ветки `master` и создать ветку `feature/#7-aims-page`
2. Сверстать страницу Целей.
3. Создать глобальный роутинг в приложении. Главная приватная страница (роут `/`) пока будет содержать две ссылки на страницу "Цели" и "Операции". А остальные страницы будут вести на главную. Будет два вида роутов - публичные и приватные.
  - a. Публичные - страницы `login` и `signup`. Их мы связали вместе в предыдущих заданиях.
  - b. Приватные - страницы целей, операций и дашборд. При попытке зайти на эти страницы не залогиненным пользователям приложение должно редиректить на страницу логина. Страница дашборда будет пока пустой
4. Создать компонент цели и прокручиваемый список этих целей. Загружать цели с сервера. Сортировку пока отложим. Компонент выбора сортировки делать пока не нужно. Карточки целей тоже пока не кликабельны.
5. Сверстать модальное окно создания цели. Запрос должен работать, а список целей - обновляться после добавления новых целей.
6. Блок **Аналитика** будет с нулями
7. Отправить `merge request` в `master` наставнику.

# Задание 8: Сортировка и аналитика

*Улучшим список целей и включим аналитику*



Список целей и добавление новых целей работает. Добавим возможность сортировать цели и соберем данные в одну кучу (раздел Аналитика).

Данные в аналитике нужно будет посчитать прямо в браузере. Прямо на основе списка всех загруженных целей.

Также добавим компонент для сортировки целей. Это будет выпадающий список с параметрами сортировки.

## Ход работы:

### Часть 1

1. Отпачкуемся от `master` в ветку `feature/#8-aims-sorting`
2. Добавим компонент выпадающего списка сортировки. По умолчанию сортировка будет "Сначала новее". Изменение сортировки, очевидно, должно перерендерить список в новом порядке.

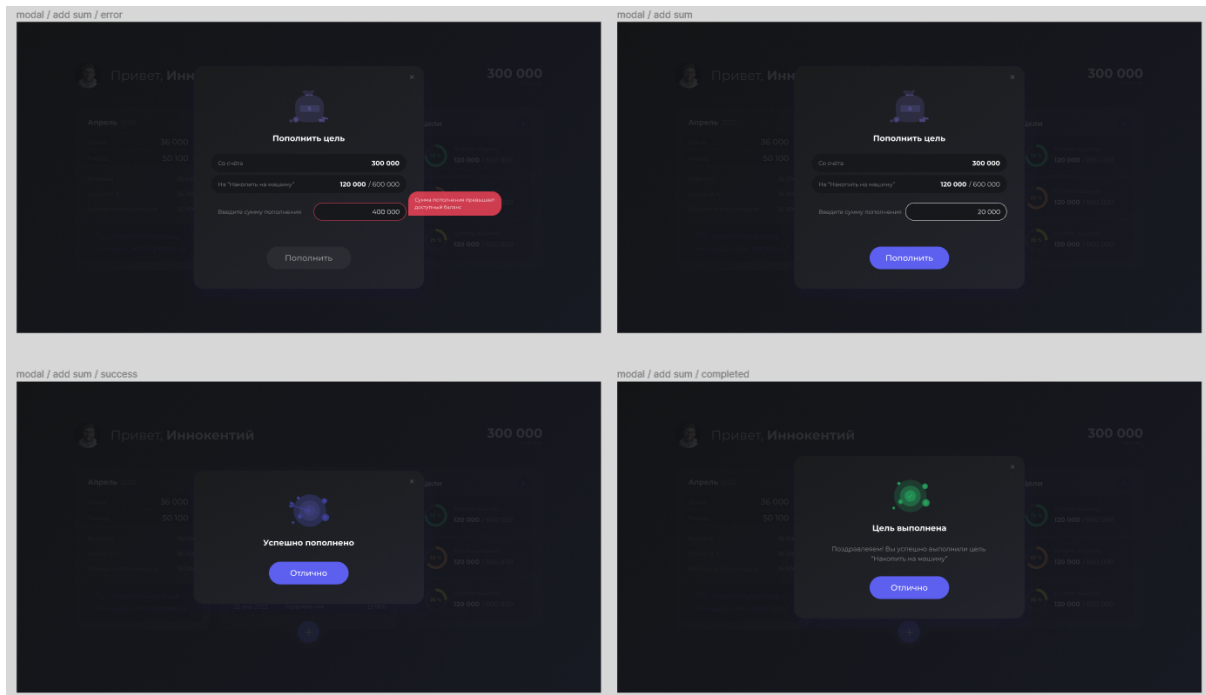
### Часть 2

1. Создаем от `master` ветку `feature/#8-aims-analysis`
2. Посчитаем данные для аналитики. Текущий месяц - это тот месяц, который выдаст **new Date**. Прошлый, соответственно, это **getMonth() - 1**.

Отправить merge request'ы в `master` наставнику.

# Задание 9: CRUD (create, read, update, delete)

**CRUD по целям. Операции с целями и состояния (Создать, Редактировать, Удалить, Пополнить, Завершить, Ошибка)**



Создадим возможность работать с целями. Редактировать цель и удалять ее. Вносить средства и выносить.

Для этого должны быть соответствующие модальные окна.

Также системы целей есть свои ивенты. “Цель удалена”, “Успешно пополнено” и “Цель завершена”. Это должно иметь вид модальных окон. Появляться должны после соответствующих действий.

Логика целей:

Цель создается с **начальным балансом** (может быть нулевым). Создание цели создает, также, **операцию в текущем месяце** на снятие средств и “замораживает” эти средства на созданную цель. **Баланс счета**, что логично, уменьшится. Операции с целями - это особые операции, их нельзя удалить и редактировать.

Если попытаться создать цель с начальным балансом, не имея денег на **счете аккаунта** - будет ошибка, которую нужно отобразить на модальном окне.

Пополнение цели работает, в целом, похожим образом. Ошибка пополнения также должна отображаться на модальном окне, чтобы пользователь понял, в чем дело.

**ВАЖНО!** После каждой операции с целями нужно перезапрашивать список операций и счета аккаунта. Он же обновится.

## Ход работы

### Часть 1

1. Создать от `master` ветку `feature/#9-aim-crud`
2. Создать модальное окно Деталей - там подробная информация о цели. Клик на цель и на пункт "подробности" в списке действий карточки целей будет вызывать это окно. Это модальное окно имеет два состояния: Просмотр и Редактирование. Можно покреативить и придумать переход между этими состояниями.
  - a. В состоянии Просмотра окно имеет вид списка.
  - b. В состоянии Редактирования окно должно изменить дизайн и кнопки действий
3. Создать модальное окно удаления цели. Вызывается при клике на пункт "Удалить". Должно давать пользователю возможность отменить свое решение и окончательно удалить цель.
4. Успешное удаление будет вызывать модальное окно состояния "Цель удалена"
  - a. Удаление цели возвращает все деньги на **Счета аккаунта.**
5. Кнопка "Завершить" цель будет вызывать модальное окно "Цель завершена"
  - a. Завершение цели возвращает все деньги на **Счета аккаунта.**

### Часть 2

1. Создать от `master` ветку `feature/#9-aim-payments`
2. Создать модальное окно добавления платежей
3. Если ввести в поле суммы пополнения число больше счета аккаунта - должно появляться предупреждение. Если пользователь все равно попытается отправить этот запрос - вывести модальное окно "Ошибка пополнения"
4. Прочие ошибки, которые будет отправлять сервер в ответ на попытку пополнения цели, будут вызывать модальные окна "Ошибка пополнения".

Отправить `merge request`'ы в `master` наставнику.

# Задание 10: Проектирование дашборда

Мы создали 2 основные страницы с данными. Пользователь может вести свои доходы и расходы, а также ставить цели и трекать свой прогресс.

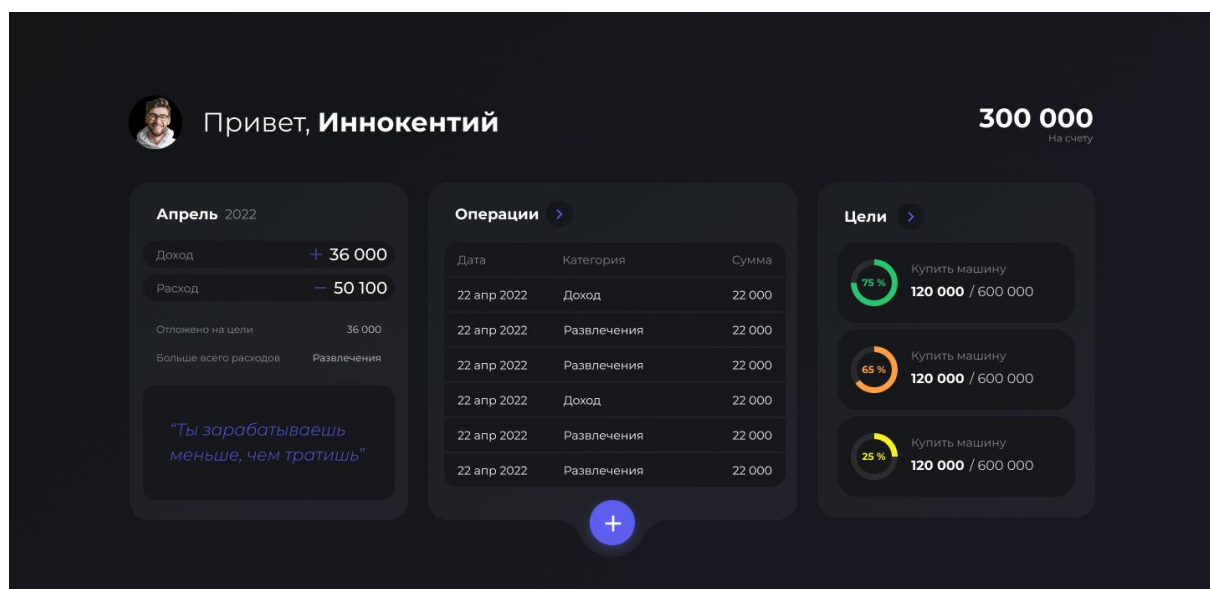
Финальным штрихом приложения станет дашборд. Здесь мы соберем ключевую информацию по текущему месяцу.

Дашборд — то стартовая страница с возможностью “быстрого доступа” ко всем ключевым функциям приложения. Если пользователь хочет внести расход или проверить цели - ему сюда.

Главное - экономить время и силы. Переиспользуйте компоненты, которые вы написали для приложения. Ничего страшного, если для этого их придется немного допилить. Это явно быстрее и проще, чем писать заново.

Также это благоприятно влияет на размер бандла приложения.

Пользователю не нужно будет ждать лишние секунды для загрузки страницы и страдать от утечек памяти попусту.



## Ход работы:

1. Ответимся от `master` в ветку `feature/#10-dashboard-page`
2. Для начала сверстаем страницу, расположив блоки как на макете. Не забываем переиспользовать уже созданные компоненты.
3. Как помним по предыдущим заданиям - все страницы ведут в Дашборд. В шапке этих страниц и располагается ссылка на дашборд. А вот шапка на этой странице будет отображать имя аккаунта. Ссылки в блоках Дашборда будут вести на соответствующие страницы.
4. Верстаем средний блок с операциями. Будет круто, если вы переиспользуете (если нужно - отрефакторите) для этого компоненты, использованные на странице Операций. Состояния у блока тоже будет два. Вся логика будет повторяться со страницы Операций.
5. Верстаем блок Целей. Не забудьте переиспользовать компонент с состоянием цели (кружочек с процентами) и прочие компоненты. Блок цели будет содержать 3 последние цели, близкие к завершению. В пустом состоянии кнопка Создать цель имеет тот же функционал, что на странице Цели.
6. Делаем блок аналитики. Используем данные:
  - а. Доход и Расход - данные текущего месяца.
  - б. Отложено на цели - сумма средств, перенесенная в этом месяце на счета целей.
  - в. Больше всего расходов = самая популярная категория.
7. Всегда сегодня: на дашборде всегда текущий месяц и актуальная информация. Грузим цели и операции по текущему месяцу. Аналитика также.
8. Место для успешных цитат пока оставим в пустом состоянии
9. Отправить `merge request` в `master` наставнику.



# Задание 11: Финал

## *Добавим серьезности*

“ПриКолДеСнОе поле” переехало на Дашборд. Здесь будут выводить случайные бизнес-цитаты про успешный успех от Джейсона Стетхума. Ручка на бэке. Создадим в последнем задании.

## Ход работы:

1. Ответимся от `master` в ветку `feature/#11-success-quotes`
2. Получим цитату с сервера
3. Отообразим в нужном месте
4. Отправить merge request в `master` наставнику.

**Проект готов:)**

# Приложение

Здесь будут полезные ссылки, которые помогут в разработке

## ОСНОВЫ

[HTML/CSS](#)

[JavaScript](#)

## HTML/CSS

[Semantic HTML](#)

[Flexbox](#)

[CSS Grid Layout - CSS | MDN \(mozilla.org\)](#)

[Sass](#)

## React

[Введение в хуки – React \(reactjs.org\)](#)

[RTK Query Overview | Redux Toolkit \(redux-toolkit.js.org\)](#)

[Redux Toolkit | Redux Toolkit \(redux-toolkit.js.org\)](#)

## Остальное

[Пять простых шагов для понимания JSON Web Tokens \(JWT\) / Хабр \(habr.com\)](#)

[Color Matters. 6 Tips on Choosing UI Colors. | by tubik | UX Planet](#)