**1. Клонирование главного репозитория**

git clone --recurse-submodules git@github.com:Morfinbrood/Tank-adventure.git

**2. Создание и подключение сабрепозиториев**

1. **Создаем главный репозиторий:**  
   git@github.com:Morfinbrood/Tank-adventure.git
2. **Создаем отдельные репозитории для логики:**
   * Backend: git@github.com:Morfinbrood/Tank-adventure-be.git
   * Frontend: git@github.com:Morfinbrood/Tank-adventure-fe.git
3. **Подключаем сабрепозитории:**  
   В терминале главного репозитория выполните:
4. git submodule add git@github.com:Morfinbrood/Tank-adventure-be.git backend
5. git submodule add git@github.com:Morfinbrood/Tank-adventure-fe.git frontend
6. **Фиксируем изменения в главном репозитории:**
7. git add .gitmodules backend frontend
8. git commit -m "Добавлены сабмодули для backend и frontend"
9. git push

**3. Коммитинг изменений**

* **Работа в сабмодулях:**  
  Каждый сабрепозиторий (backend, frontend и вложенные) — самостоятельный проект. Вносите изменения, делайте коммит и пуш непосредственно в этом репозитории.
* **Обновление указателей в родительском репозитории:**  
  После коммитов в сабмодулях необходимо обновить указатели во всех вышестояших сабпозиториях вплость до корневого проекта:
* cd <главный\_репозиторий>
* git add backend frontend
* git commit -m "Обновлены указатели сабмодулей до новых версий"
* git push

*Важно:* Если изменения происходят во вложенных сабмодулях (на 5 уровне вложенности и т.д.), необходимо обновлять указатели по каждому уровню, чтобы главный репозиторий ссылался на актуальное состояние.

**4. Обновление сабмодулей**

Для получения изменений, сделанных другими разработчиками:

* Выполните в главном репозитории:
* npm run update:submodules

Этот скрипт подтянет последние коммиты из всех сабмодулей (включая вложенные).

**5. Общий рабочий поток**

* **Внесение изменений:**  
  Сделали коммит в нужном сабрепозитории. Так как каждый саб модуоль полностью самостоятельная еденица, то может быть достаточно просто работать в нем и все проверив юнит тестами, сделать коммит в него.
* **Обновление указателей:**  
  Для каждой серии изменений делайте отдельный коммит в сабрепозитории, а затем обновляйте указатели в вышестоящих репозиториях до корневого проекта.
* **Принцип:**  
  – **Никаких cherry-pick’ов!**  
  – Версионность каждого модуля отражается через отдельные коммиты.  
  – Если потребуется передача данных через несколько уровней, это сигнал к пересмотру архитектуры, либо нужны обоснования расширения функционала на каждом уровне.

*На каждом Pull Request опишите, что и почему изменено, а также зафиксируйте обновление версии модуля в package.json.*

Эта инструкция поможет сохранить прозрачность и независимость модулей, а также обеспечит строгую версионность от общего к частному.

Примечение:  
*Это выглядит тяжело, особенно, если 1-му писать проект, но плюсы:  
1. Версионность – у нас будет отдельная версия рут проекта на каждую фичу, для тестов именно ее. И все плюсы версионности.  
2. Если надо будет провести какую-то зависимость или переменнную через какой-то сабмодуль, то это будет четко видно. А идея проекта не прокидывать зависимости. Т.е. если нам надо какую-то переменную к примеру получить, через 2 сабмодуля, то надо отдельно писать логику по ее полной поддержку и обработке в каждом модуле и обосновывать ее там нахождение. ИДЕЯ в том, чтобы каждый модуль быть самостоятельным и никаких зависимостей сквозь модули быть не должно. Если таковое понадобиться, то значит есть архитектурный просчет и надо думать как архитектурно оправдать добавление данной функциональности на каждом модуле. Пока что я искренне верю, что смогу это решать и не переплести проект взаимными зависимомстями и передачей объектов сквозь другие модули просто потому что по-другому никак.  
  
Поэтому я хочу затруднить разработчикам вермишеллить код, ухудшать его прозрачность, зависимости строго от общего к частному, никак иначе. И проверить на практике как это будет работать.*