This proposal concerns the financing of the ongoing development of **smoldot**.

Overview: what is smoldot?

This section contains a general presentation of the smoldot project and is present in every smoldot treasury proposal. If you are already familiar with smoldot and its purpose, I invite you to skip directly to the milestones delivery section.

A blockchain, such as the Polkadot blockchain, consists in two things: a certain state (e.g. a list of accounts), and a peer-to-peer network. There exists only two ways to access the state: either run a *client software* (such as the client software made by Parity Technologies) that connects to this peer-to-peer network, or rely on something else running that client software for you. In both cases, this client software is critically important.

Currently, most blockchain-related web applications or UIs connect to a trusted server (such as with Infura or so-called JSON-RPC nodes) that runs the client software and acts as an intermediary between the web application and the actual blockchain. In other words, the web server "runs the client software for you". This trusted server is a considerable security issue: it can be hijacked and send bad information to the web application, can go down and leave the web application non-functional, etc. Additionally, running these trusted servers incurs significant complexity and maintenance costs. Getting rid of these trusted servers is **one of the primary objectives of building a decentralization blockchain**.

The smoldot light client is a client software capable of connecting to the peer-to-peer network. Compared to the official client, the smoldot light client intentionally provides fewer capabilities (it cannot author blocks, vote for finality, or easily look at the history of the chain) but is considerably lighter in terms of CPU, memory, and disk space consumption. Because it is so light, the smoldot light client can be embedded within a web page in order for this web page to establish a direct connection to the peer-to-peer network of the chain. This eliminates the need to rely on a trusted server, and thus eliminates the security issue and maintenance costs.

The smoldot GitHub repository can be found here: https://github.com/smol-dot/smoldot

I invite you to read the README of the GitHub repository if you are interested in more details.

History and team

The smoldot project started in December 2019 by me, Pierre Krieger, the submitter of this proposal, and is presently still driven entirely by me.
The project was initially started within Parity Technologies. Since February 2023, it has been financed through treasury proposals.

Pierre worked for Parity Technologies for roughly 5 years, between 2017 and 2022. At the time of writing of this text, he is the third biggest contributor to Substrate with 623 pull requests and the biggest contributor to rust-libp2p with 640 pull requests. He initially started and led the peer-to-peer networking team in addition to contributing to other parts of Substrate (e.g. HTTP requests in offchain workers). Pierre is likely one of the most knowledgeable people when it comes to how the Polkadot client works.

Before and during his time at Parity Technologies, Pierre has also led several other open source projects, such as vulkano (3.6k GitHub stars), glium (3.1k GitHub stars), glutin (1.8k GitHub stars), or redshirt (1.4k GitHub stars).

- GitHub profile: https://github.com/tomaka/
- E-mail address: pierre.krieger1708@gmail.com
- Matrix account: @tomaka17:matrix.org
- DOT address (for anything smoldot-related): 15kgSF6oSMFeaN7xYAykihoyQFZLRu1cF5FaBdiSDHJ233H5

At the time of the writing of this text, the smoldot repository contains 133k lines of Rust source code. Despite its size and the overall complexity of writing a client implementation, the source code is overall well organized, of good quality, reasonably well documented, and no large-scale refactoring is likely to ever be needed. It is unlikely for any major blocker to be encountered in the future with regards to implementing a specific feature. Additionally, the list of open issues is actively maintained, and the size of the backlog is reasonable.

One of the main issues that smoldot could potentially face is the fact that it is being built only by one person. However, it should be considered that the code of smoldot is reasonably clear and reasonably well documented, making it relatively easy for someone else to take over if it ever becomes necessary. Furthermore, smoldot is nothing more than source code, licensed under GPL3. Anyone can legally fork the code and continue working on it themselves.

Delivery of the previous milestones

This is not the first proposal for the financing of the development of smoldot. **The previous proposal can be found here.**

The table below recapitulates the milestones from the previous treasury proposal and presents the work that has been done. Keep in mind that work is still happening as this proposal is being discussed, and as such the table below might be missing some items.

Title and work done	Remarks
General maintenance of the project Too many changes to list reasonably, see https://github.com/smol-dot/smoldot/commits/main?branch=main for the full list of commits. I also invite you to look at the CHANGELOG. Keep in mind, however, that the CHANGELOG doesn't cover internal changes.	Noteworthy changes include updating to changes in the JSON-RPC specification, better loading times, and another refactoring of the JSON-RPC server. The full node has also seen significant changes, but note that the hours I have spent working on the full node weren't part of the proposal.
Try add support for running smoldot in a web worker/worker thread #499, #489, #491, #494 (main proof of concept), #504, #505, #511, #517, #524, #528, #529, #532, #538	Smoldot can now run in a web worker/worker thread by doing the execution in the background and sending/receiving messages with the foreground. As suspected, support for multithreading can't be done yet because it is blocked by third parties (Firefox and Rust), but smoldot is ready to add support very quickly once unblocked.
Implement support for child tries #631, #639, #669, #670, #673, #678, #680, #684, #743, #752, #763, #722	Smoldot now supports child tries! As expected, this was quite complicated due to the full node prototype getting in the way, and due to the lack of documentation about how child tries work.
Cache the runtime code of the chain in order to speed up connecting to a chain #863	Done! Here again smoldot was blocked by third parties taking more time than expected. In order to be unblocked, I ended up implementing this feature in a sub-optimal way that might lead to more cache misses.

Project objectives, status, and direction

In the last three months, smoldot has been integrated in two additional note-worthy projects (that I know of):

- SubXT
- Cumulus (parachain full nodes can use smoldot in order to connect to the relay chain)

It was furthermore **mentioned in several talks of Polkadot decoded, and was the main topic of one of the talks**.

The smoldot project is following five main high-level objectives: being a correct and secure implementation, providing a good end user experience, improving the infrastructure developers experience, improving the developers' experience, and building a new full node implementation.
Let's take a look at the status of each of these objectives, and how smoldot plans to fulfill them in the long term.

Being a correct and secure implementation

The most important objective of smoldot is to conform to the Polkadot protocol.

This objective is as a whole generally fulfilled. Unfortunately, it is sometimes hard to know whether smoldot behaves as it should because the official specification is still very incomplete. The recent creation of the RFCs repository is however a good step towards formalizing changes to the specification.

It would be desirable for smoldot to undergo an audit in the future. **I would be happy to collaborate with an auditing company that would be willing to audit the source code of smoldot**. I am also, more generally, happy to answer any technical question concerning aspects of the source code.

Providing a good end user experience

While end users are normally not *directly* using smoldot, they are using software that relies on smoldot. For example, the PolkadotJS UI or the staking dashboard both have an option that allows using smoldot to connect to the chain. The ultimate end goal is for all Polkadot/Kusama/… UIs and all parachain UIs to use a light client such as smoldot.
As such, if smoldot is slow, the UI is slow as well.

After the latest changes (#529, #532), users of smoldot now **have the possibility to run it in the background**. This significantly improves the user experience by removing the small stutters that browsers experience when running smoldot in the foreground.

An important part in improving the user experience is the design and integration of **the new JSON-RPC API**. Early June, I opened a forum thread in order to provide an overview and ask for feedback. I have since then received good feedback, and my general opinion is that the API is very close to being stabilized.

Now that #92 has been done, the smoldot initialization no longer downloads 1.2 MiB every single time it connects to a chain. This saves a few hundred milliseconds depending on the speed of the end user's connection.

The time it takes between the start of the smoldot Rust code and the end of the warp syncing is around 850ms on my machine, assuming an up-to-date database. Around 50ms to 100ms could be saved by using WebRTC instead of WebSocket as the protocol to connect to the network (currently blocked by Substrate), and another 100ms to 150ms could be saved by doing some refactoring to the warp syncing code of smoldot.

Improving the infrastructure developers experience

The objective of smoldot is to eliminate the need for JSON-RPC nodes. This has the side effect of removing the burden of having to deploy and maintain these JSON-RPC nodes for the teams building parachains.

Nothing much has changed compared to the previous proposal. Smoldot is currently still waiting for the WebRTC feature to be implemented in Substrate. As explained in the previous proposal, the end goal for smoldot is to be able to “magically” connect to any parachain without any manual intervention.

Improving the developers’ experience

Smoldot has re-implemented many Substrate features, sometimes in a way that makes them more simple to use externally.

For example, support for child tries has recently been added to smoldot, allow the Ink! team to use chopsticks to test contracts.

If you are a developer in need of specialized tools that Substrate doesn’t provide, or that Substrate does provide but in a way that is too difficult to use, feel free to open an issue or discussion in the smoldot repository.

As an example, such a tool could be a small program that connects to a specific IP address and port, and prints the PeerId and software version of the Substrate node that it finds.

Building a new full node implementation

While most of the development of smoldot focuses around its light client, the smoldot repository also contains a prototype of a full node built upon the same primitives that the light client uses.

Having multiple functional full node implementations would be a good thing for the Polkadot network, as it would reduce the risk that the network collapses in case of a bug or security issue in the unique client.

Previously, very little effort was spent on improving the smoldot work-in-progress full node implementation. In this proposal and future ones, however, I would like to focus a bit more on it. While implementing a relay chain validator would be a very difficult endeavor for a single person, implementing a full node (non-validator) or a parachain collator is a realistic long-term goal.

Quarterly milestones and budget

Based on the directions laid out in the previous section, work items for the next three months have been picked. The choice of these work items was done based on what seems to me to

be the highest priority. However, **I am totally open to feedback if you think that priority should be put somewhere else**.

This proposal covers three months of work, from August to October, after which a new proposal will be submitted.

An hourly rate of 300 € is applied.

General maintenance of the project: - Fix panics and/or security issues that may arise - Fix quality of life issues - Fix overly-high computational complexity issues - Improve the project-wide documentation and code readability - Keep dependencies up-to-date A lot of changes are either unpredictable at the moment, or are too small to be their own milestone. See the previous milestone delivery for examples.	80 hours	24 000 €
Research and implement a proper discovery and peering process The strategy for discovering nodes and choosing which peers to connect to is currently not very refined. It turns out that this is a difficult problem for a variety of reasons. For this reason, it is currently not clear whether smoldot is always capable of reconnecting after an Internet connectivity outage.	60 hours	18 000 €
Refactor and fix all the remaining issues in the warp syncing code Several open issues (#864, #67, #119, #131) require changes to the warp syncing code. The warp syncing code has gone through many iterations in the past and is suffering from technical debt. It is time to refactor it and fix all the open issues.	60 hours	18 000 €
Implement the new JSON-RPC and the most-commonly called legacy JSON-RPC functions in the full node This is a step towards having a usable full node.	40 hours	12 000 €
Total	240 hours	72 000 €

The exact amount in DOTs will be calculated when the proposal is submitted on chain using the 7-days average found on subscan and the current USD <-> EUR exchange rate found on xe.com. The destination address is **15kgSF6oSMFeaN7xYAykihoyQFZLRu1cF5FaBdiSDHJ233H5**.

Please note that these milestones are provided as a best effort estimate, and the reality might differ. This proposal assumes a certain level of trust, and an emphasis is made on code quality rather than delivering the milestones at any cost. The actual work that has been performed will be showcased in the treasury proposal asking to fund the next 3-months period. If you were to be unsatisfied with my work, I am open to discussing the way I focus my efforts.